ICL DAP (1)

Regeneration from IFS codes of highly compressed natural images.

S. F. Reddaway

Changes from Issue 1

The second half of the algorithm is quite different; essentially sorting and other work is replaced by a scalar write of the image. This produces a fivefold performance improvement; in the example I have doubled the number of points and more than halved the time.

The Algorithm

The Iterated Function System (IFS) codes of images (see Barnsley and Sloan, Byte p125, Jan 1988) are of the form:

W	a	b	c	d	e	f	p
1	a1	b1	c1	d1	e1	f1	p1
2	a2	b2	c2	d2	e2	f2	p2
3	a3	b3	c3	d3	e3	f3	p3
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
n	an	bn	cn	dn	en	fn	pn

The p's are probabilities and must add up to 1 for the complete set of n "affine transformations". n can be be arbitrarily large. This note applies to black and white images, but the techniques "can be extended to grey and colour".

An arbitrarily long series of points is generated from the table by the following algorithm:

$$x(i+1) = a(j)*x(i) + b(j)*y(i) + e(j)$$

$$y(i+1) = c(j)*x(i) + d(j)*y(i) + f(j)$$

where j is chosen (for each new point x, y) to be a number from 1 to n with probabilities given by the pj. The values for each transformation have been chosen to make the x, y points remain bounded (contractive transformations). The first dozen points are thrown away (x(o) and y(o) are unimportant and can be taken as o, o), the remainder are plotted.

Implementation on DAP

For big complex images with many points, a "vertical" mode DAP implementation seems best. Each PE generates a string of points using a global table and the Monte Carlo selection of transformations is also global. To prevent each PE generating the same string of points, each PE decides locally whether to accept an update. I suggest a local probability of 3/4 or 7/8 would be suitable, a compromise between ensuring that duplicate strings are broken up and "wasting" points.

The multiplications are now scalar-matrix (about 12-bit fixed point would be suitable - in Fortran Plus 24-bit floating point for convenience), and strings of points can be rapidly generated with "skipped" points being marked. 680 points/PE might be a suitable number if the local acceptance probability were 3/4, with the next step being a compression to squeeze out the "skipped" points. This will leave about 512 points in each PE; any surplus over 512 can be ignored. The points then have the bits of their x and y values interleaved with an eye on the desired mapping of the final image. Bits may also be dropped ; e.g. a 1K by 1K image needs only 10 bit x and y. These 20-bit numbers are now converted to horizontal mode and used serially to write the image. This is fastest if a complete row of memory is used for each pixel, but if memory is insufficient (a iK x 1K image would need an 8M byte DAP) a fraction of a row can be used. All the image space is initialised to one polarity, and then, for a black and white image, pixels are written to the opposite polarity; a grey image accumulates a pixel intensity. Finally the image data is moved (if necessary) to a mapping suitable for output.

Performance (low level coding)

Random number generation (fast algorithm)	12 cycles/point matrix
Generate new matrices of points	600 cycles/point matrix
690 iterations	0.4 x 10**6 cycles
Compress 20 bit numbers @ 25 cycles/BIP	0.34 x 10**6 cycles
Write black & white image @ 5 cycles/pixel	2.5 x 10**6
Image remapping	0.2 x 10**6
TOTAL	3.5 x 10**6 cycles or 0.35 seconds
(Fortran-Plus estimate	2 to 3 seconds)

Barnsley and Sloan quote 30 minutes for reconstructing complex colour images on a Masscomp 5600 workstation, and several frames

a second on their IFSIS prototype hardware; these figures may not be for comparable images, nor comparable with the example above.

Grey Images

Two adaptations. First, duplicates are summed .Second, a transformation could have a "weight" (4 bits?); this would increase the record size of a point (20 to 24 bits?). Speed is estimated at 30% slower.

Colour

There may be better ways, but the best I can guess for colour is to have 3 grey images to give the 3 colour values. Sparse data may be improved by smoothing over a neighbourhood of pixels.

Image Compression

Barnsley & Sloan have not given details on the compression side. However, the approach may be an iterative one with compression parameters being adjusted to give better fits to the input image. In this case, the core of the processing may be repeated use of the regeneration algorithm. A figure of 100 hours on the Masscomp is mentioned for compressing complex colour images.

Initial Distribution:

UK:	USA:
GM	BA
MRH	WT
CW	RTh
CP	KI
DJHu	
PMF	
DP	
BT	
JQ	
AGB	
AW	

THE DAP AS A FILESTORE SEARCH ENGINE

R.M.R. Page and S.F. Reddaway.

Abstract

This paper examines the performance achievable with the current generation of DAPs on character file searching tasks, and considers the implications of the direct i/o facilities now being provided with these machines.

Introduction:

A number of investigations [1] [2] [3] have been carried out into the suitability of array processors for retrieving information from disc files, but until recently the Distributed Array Processor (DAP) [4] [5] has had to rely on the i/o system of its host computer, and it has consequently been impractical to use it for this purpose. The introduction of a Fast Input Output (FIO) facility to the current generation of DAPs has altered the position however, and the opportunity now arises to connect discs, through the FIO interface enabling data to be transferred directly into or out of the DAP array. A standard disc drive could be connected in such a system, but to fully exploit the search rates achievable with the DAP, a number of high performance drives transferring data in parallel across the interface would be needed.

The current version of the FIO can transfer data at approximately 40 MB/s, and this paper investigates the rate at which a 32 * 32 DAP can carry out both relational and exact match comparisons between this data and query keys, and compares the rate with that of FIO transfers. Both files of formatted records and text are considered.

Code to perform search algorithms was written in the DAP assembly language APAL. FIO hardware was used to create data files in the format described below.

Test results presented here were achieved using the ICL DAP-2 which has a machine cycle time of 160ns. The AMT* DAP-510 has a cycle time of 100ns, so search rates would be proportionately increased. This latter machine provides for connection to a fast digital highway allowing even higher data transfer rates into and out of DAP memory.

File Format:

Data is searched fastest if it is held in "vertical" format with many consecutive bytes held in the same Processing Element (PE). In the block format used in this work, 14 consecutive characters are held in the memory of each PE. The start of records and fields within variable record length files would normally be located by serial hops along the data. While this is feasible, an alternative DAP specific format has been used for this study.

* Active Memory Technology Ltd, Reading, U.K.

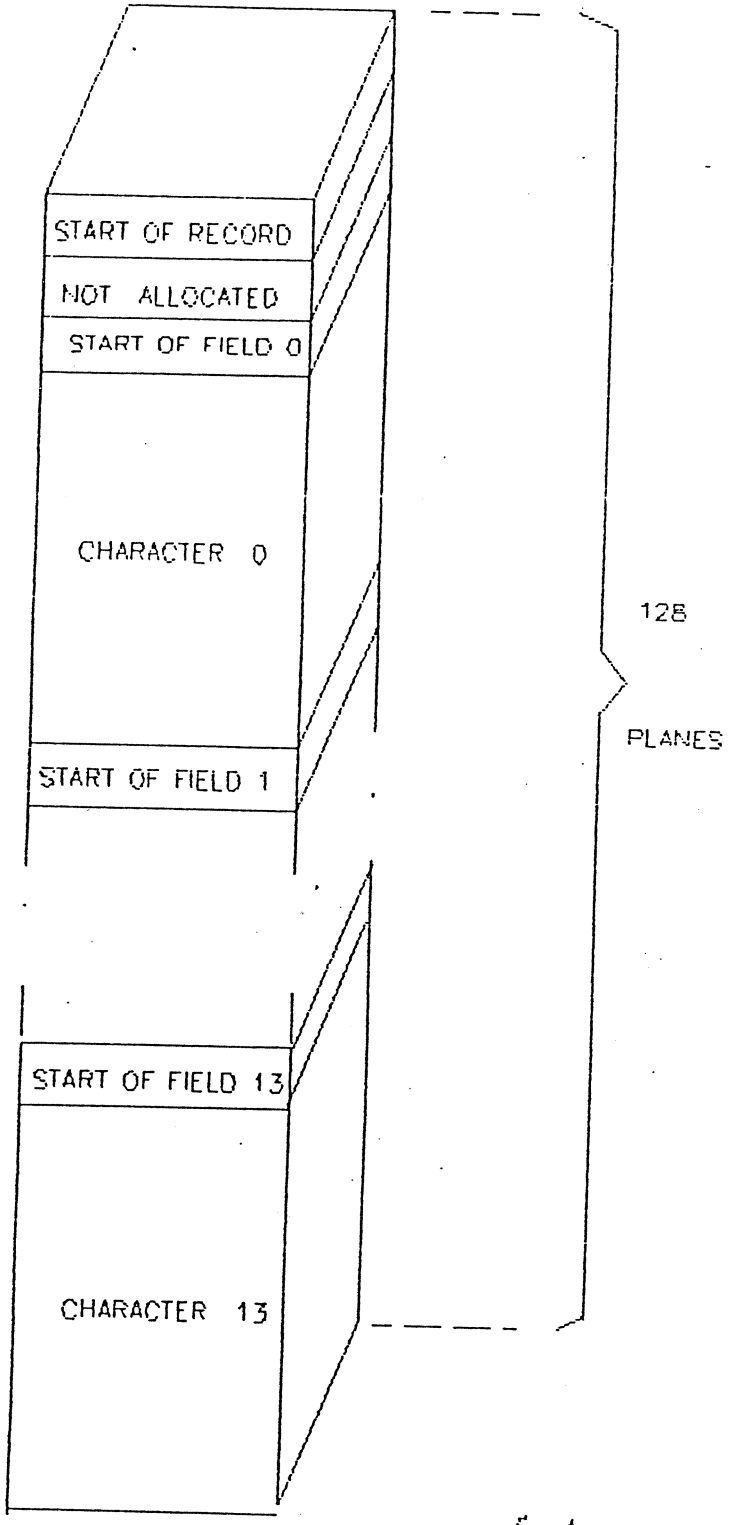


fig 1.

Records start only at the first character position in a PE and are preceeded by a one bit field which is set at every point in the plane where a start of record occurs. Each character is also preceeded by a 1-bit field which is set at the points where a start of field occurs (see fig 1). Records continue from one PE to the next PE in the row, then to the neighbouring row and so on. They are not permitted to cross block boundaries. With the extra logical planes specified, a block occupies 128 planes of the DAP array, of which 127 planes are used.

The first byte in each field is a field type byte Thus all fields at a particular start plane other than fields of the type being searched for will be eliminated by the first byte comparison. Dependent on the data, this may also mean that some field start planes are eliminated by a single byte comparison, while others may be eliminated even sooner if there are no active field start points in that plane.

Performance Measurement:

Machine cycles per bit plane (c/bip) are a convenient means of comparing relative processing loads in performing the various stages of a search operation. A bip comprises 1 store bit at the same address in the memory of each PE in the array. For example in a 32 * 32 array, a bip comprises 1024 one-bit elements. The processing load in terms of c/bip can readily be translated into a search rate of MB/s. For work done on a block of data consisting of 128 planes in which 14336 characters are stored, if the machine cycle time is 160 ns, then 5 c/bip is equivalent to 640 cycles/block or a search rate of
$(14336 * 10^3)/(5 * 128 * 160) = 140$ MB/s.

Achieving the Required Format:

If data is held on discs in horizontal format, it can be read by other systems, provided allowance is made for the extra one-bit

fields. With appropriate hardware such as the FIO of DAP-2 vertical format can be achieved with no additional i/o time penalty, the transformation being performed on the fly as data is moved into the DAP array from the disc system. Alternatively horizontal data when first input to the array can be put into the required format using parallel data transforms [4] and stored on disc in vertical format for subsequent use.

Search Method:

Search keys are held in horizontal format in successive rows of a plane. From there they are loaded into a machine register and a bit by bit comparison carried out with data held in successive array planes from the appropriate start point. The processing cost for the actual comparison part of the operation is only 1 c/bip for exact match tests and 3 c/bip for relational tests.

For each start plane a test of the search key against the data strings is started and continues until all strings have failed, or until the end of the key is reached. First the start plane is tested to ensure that there is at least one active start_of_field in that plane. If this test is positive, the start_of_field plane is used as an initial mask. It is AND'ed with the result of byte comparisons, and total failure is then tested for after each character. When the last character in each PE has been tested, the mask must be moved to the neighbouring PE's and the comparison continued. For fixed length records, all fields of a particular type start in the same plane. Thus only one start plane needs to be searched. This is also true for the fixed length fields in variable length records, provided that the variable length fields are kept at the ends of the records.

Relational comparisons are achieved by first carrying out an exact match test until all strings have failed or until the end of the key has been reached. The result of the exact match is then thrown away and the relational test is carried out from this point back to the start. Where the relational comparison moves from one set of PE's to another, the plane of carry bits must be adjusted in the opposite direction to that used for the exact match mask. The resultant mask is finally AND'ed with the result of the match on the field type byte.

Search Performance:

A number of factors affect the rate at which the DAP can search data:

(a) The number of active start_of_field planes.

(b) The search key length and comparison type. A test for an exact match can be carried out faster than a relational test.

(c) Number of search criteria. If the required condition is the AND of separate test results on a number of criteria, and if one of these tests results in failure everywhere, it is possible to eliminate all records in a block without performing all of the tests. Otherwise the tests must be carried out sequentially on that block until a result is obtained.

(d) Whether fixed or variable length records are being searched.

For this work a data file was constructed such that there were

active start_of_fields in each start plane. This is a worst case situation and would not be true for all data files.

Tables 1 & 2 show how the DAP block search rate involving a single key comparison varied under specific conditions. The rates will to a certain extent be data dependent. The figures given are measured; the coding could be improved to increase performance by at least 20% for exact matches and by 6% for relational tests.

TABLE 1 - Search Rates With Variable Length Records

TEST		Number of characters exact match part of test lasts for in 1 start posn.	1 or more hits in one start position	
			Search Rate MB/s	c/bip
Valid Field Type Bytes in 1 Start Plane.	Exact match. 15 character key	-	131	5.3
	Exact match. 8 character key	-	159	4.4
	Exact match. 3 character key	-	184	3.8
	Relational test. 15 character key	15	90	7.8
	Relational test. 8 character key	8	124	5.6
	Relational test. 3 character key	3	163	4.3
Valid Field Type Bytes in 3 start Planes.	Exact match. 15 character key	-	123	5.7
	Exact match. 8 character key	-	147	4.8
	Exact match. 3 character key	-	168	4.2
	Relational test. 15 character key	15	82	8.5
	Relational test 8 character key	8	108	6.5
	Relational test. 3 character key	3	137	5.1

Note: These figures are for the ICL DAP-2. Search rates on the AMT DAP-510 should be approximately 1.4 times higher.

If fixed length records are being tested, then only one start_of_field plane will be relevant to the search. Thus higher search rates are possible as indicated in table 2.

TABLE 2 - Search Rates With Fixed Length Records.

TEST	Number of characters exact match part of test lasts for in 1 start posn.	1 or more hits in one start position	
		Search Rate MB/s	c/bip
Exact match. 15 character key	-	273	2.6
Exact match. 8 character key	-	434	1.6
Exact match. 3 character key	-	693	1.0
Relational test. 15 character key	15	144	4.9
Relational test. 8 character key	8	257	2.7
Relational test. 3 character key	3	497	1.4

Note: These figures are for the ICL DAP-2. Search rates on the AMT DAP-510 should be approximately 1.4 times higher.

Failure Statistics:

If data were purely random, then using an eight-bit character code, the probability of any individual byte match failing would be $1 - 1/256$, and the probability of a match on a pair of bytes failing everywhere would be $(1 - 1/256^2)^{1024}$. However character data is not random. If we assume that the probability of a match failing is say 19/20, then table 3 indicates the probability that an exact match test will fail everywhere after 2,3 and 4 characters. No data will have these exact properties, however the overall effect of quite large changes in the probability assumed, is not very great.

TABLE 3 - Probability of Exact Match Test Failing in All PE's

Number of bytes used in exact match test	2	3	4
Probability of exact match failing in all PE's	8%	88%	99%

These probabilities show that, the 3-character key entries in Tables 1 and 2 are the most important ones in indicating how quickly total failure will be established from individual start planes, irrespective of key length.

Text:

Text can be searched by this method, though the code adaptations have not been implemented. Text is held as text and character strings are searched for. This is quite different from the method of reference 3, in which words in a document are hashed into a fixed length binary array, and all word order is lost. The latter can be very powerful and can be performed on the DAP in the same way as on the Connection Machine; but it is not the method described here. Reference 2 describes a somewhat similar algorithm to reference 3 implemented on the DAP.

Start_of_field bits are not relevant to text searching and would therefore be omitted. Failure statistics may not be so favourable as for the formatted record case; if we assume that failure in every PE would not occur on average until 4 or 5 characters had been compared, then a search rate for a single key comparison of about 60 MB/s can be inferred from table 1, after allowing for active starts existing on every character plane.

Quorum functions in which m out of n search criteria must be satisfied, are frequently applied in text searching. The necessary operations can be programmed on the DAP at relatively small cost. If we take for example a requirement to match 5 keys out of 6, the statistics are complicated but a typical block is likely to fail after about 3 keys have been tested. This suggests a search rate of about 20 MB/s would be achieved.

Elastic matching has been studied theoretically. Here there may be substantial gaps of unknown lengths between fragments of strings to be matched. Good performance is possible although the implementation would be more complex.

Matching Disc Transfers to DAP Search Performance:

To achieve maximum performance, the DAP search rate should keep up with the disc/FIO system.

The search rate is data dependent, but can be smoothed by using a buffer area to hold a number of blocks of data. The degradation in search performance due to hits in individual blocks is then masked by a higher search rate in the remaining blocks.

For parallel disc systems files can be distributed across a number of drives at byte or bit level, with corresponding file fragments occupying identical physical address space on each drive. If files are not too fragmented, then the overall disc transfer rate can be improved by the transfer of large amounts of data in one sequence.

As the rate at which a system can access and search data increases, it becomes more attractive to use a relatively simple level of indexing and as a consequence to search larger sections of files. A double buffering arrangement in DAP memory can be used, one buffer being searched while the other is loaded.

For normal disc speeds (about 2 - 3 MB/s) the DAP search rate is one to two orders of magnitude faster, and searching could be a minor load in a multi-tasking environment.

Processing File Joins:

Joins between files can be dealt with using bit maps [7] [8]. A hit on the first file would result in one bit being written into a bit map. One method is to address the bit map with a coupling index that is held in the record and has been precompiled from the join field. When the second file is examined a hit causes the bit map to be read using a similar addressing method and if the bit read out is set then the join has succeeded.

An alternative to the use of coupling indexes is to hash a field of the file under examination, and use the result to address the bit map.

To reduce the effect of collisions to a reasonable level a number of bit maps should be used, each addressed by a separately derived address. One way of achieving this is to divide the field to be hashed by different prime numbers, and to use the results to address separate bit maps. When the second file is being examined and the bit maps are read, the resulting bits are AND'ed to establish whether a join has been found between the two files.

Table 4 is based on theoretical assessment. It indicates the processing load associated with each stage of processing a query involving a file join. Assumptions have been made in deriving the figures for this table that the fields to be hashed represent approximately 5% of the data, and that records average 200 bytes in length.

TABLE 4 - Processing Load Associated With File Joins.

F I L E 1.	Variable length records.		Fixed length records
	c/bip		
Search, 5 criteria, 8 character keys, relational tests.	28	28	13.5
count hits in block	1	1	1
Assumed percentage hits	95%	2%	50%
	c/bip		
For >10% hits, transpose all data to horizontal format.	35	-	35
Extract horizontal data	10	-	5
Transpose to vertical format	1.7	-	0.9
For <10% hits, extract and insert data serially	-	6	-
Hash data held in vertical format and transpose to horizontal format	9	0.2	5
Write Bit Map.	10	0.1	5
Approximate total processing load for file 1.	95	35	65
F I L E 2			
Search (criteria as above)	28	28	13.5
Count hits in block	1	1	1
For >10% hits transpose all records to horizontal format.	35	-	35
Extract join fields from horizontal data.	10	-	5
For <10% hits extract data bit serially to horizontal form.	-	6	-
Hash data in horizontal format.	30	1.5	16
Read bit maps.	6	0.1	3
Assumed final % hits	10%	0.1%	5%
Find data to be retrieved.	2	0.02	1
Assumed % of total data to be retrieved.	5%	0.05%	1%

(continued.)

Retrieve hit data from records in horizontal format.	10	-	2
Retrieve hit data from records in vertical format, storing horizontally.	-	2.5	-
Approximate total processing load for file 2.	122	39	76

It is advantageous to count the number of hits after the initial searching. If there are more than about 10% hits, it is better to transpose the whole data to horizontal form in order to make the extraction of the join fields more efficient; 32 bits are extracted at a time instead of 1. For lower hit rates, the hit join fields are extracted serially.

For file 1 it is not practicable to save data that may potentially need to be retrieved; a second pass would be needed. It is therefore possible to store these fields vertically, and to continue building up a set of join field data until either the file being read is finished, or a set of 1024 hits is completed. Hashing is done in parallel on this vertical data which is then transposed to horizontal format for writing the bit map.

If there is data to be retrieved in file 2, then either the file is scanned twice, or hashing is done before the data to be retrieved is overwritten. In table 4 the latter is assumed and, because of the lower parallelism, hashing is done on the data held in horizontal format.

Table 4 assume the use of three separate bit maps to reduce the effects of collisions. The middle column shows an average load of about 37 c/bip or a processing rate of 19 MB/s.

Dealing With Hit Records:

Where file join processing is not involved, hit records are extracted by similar methods to those described in the previous section, the precise strategy again depending on the percentage of the total data that produces hits.

Conclusions:

Now that direct i/o facilities have been provided and data can be moved rapidly into and out of DAP memory, utilising the DAP for high speed searching of files becomes a very attractive option. Single disc drives can be connected and the DAP used in a multi-tasking environment, dealing with a number of different data base queries. Alternatively a system of parallel discs can be connected transferring data at peak rates approaching the FIO transfer rate.

Search rates will normally exceed the rate at which data is loaded into DAP memory. For simple queries, measured DAP search rates were in the range 80 - 700 MB/s, which exceeds feasible disc transfer rates. For complex queries involving joins and high hit rates, DAP search rates exceeded 10 MB/s.

Acknowledgements:

R.M.R. Page would like to thank Prof. Dennis Parkinson for help and facilities provided at the DAP Support Unit, Queen Mary College, London and David Hunt (AMT Ltd.) for explaining the FIO system and for arranging machine time at AMT.

REFERENCES:

1. P.W. Williams - Using the Distributed Array Processor for Information Retrieval. IUCC Bulletin, 1, 1979, pp 7 - 10.

2. A. Pogue & P Willett - Use of Text Signatures for Document Retrieval in a Highly Parallel Environment. Parallel Computing, 4, 1987, pp 259 - 268.

3. C. Stanfill & B. Kahle - Parallel Free Text Search on the Connection Machine System. Communications of the ACM, 29, 12, 1986, PP 1229 - 1239.

4. P.M. Flanders, D.J. Hunt, S.F. Reddaway, D. Parkinson - Efficient High Speed Computing with the Distributed Array Processor. Reprint from High Speed Computers and Algorithm Organisation, 1977. Academic Press Inc.

5. D.J. Hunt and S.F. Reddaway - Distributed Processing Power in Memory. "The Fifth Generation Computer Project" (ed G.G. Scarrott), Pergammon Infotech Ltd., Maidenhead 1983.

6. P.M. Flanders - A Unified Approach to a Class of Data Movements on an Array Processor. IEEE Trans on Computers, C31, 9, Sept 1982, pp 809 - 819.

7. V.A.J. Maller - The Content Addressable File Store CAFS. ICL Tech J, 1, 3, 1979, pp 265 - 279.

8. E. Babb - Implementing a Relational Database by Means of Specialised Hardware. ACM Trans on Database Systems, 4, 1, March 1979, pp 1 - 29.

ABSTRACT FOR CONPAR88

Achieving High Performance Applications on the DAP

S.F. Reddaway

Techniques for achieving high performance on the DAP architecture will be discussed and illustrated by identifying the key performance factors in a range of applications and algorithms. These will be examples the author has been involved with implementing, drawn from the following areas:

Application Areas

- Signal Processing
- Image Processing
- Physical Simulation
- Number crunching
- Number theory
- Data correlation/information retrieval
- Graphics
- Pattern Recognition

In some applications outstanding performance is achieved, unmatched by any other machine. In others, the appeal is a very competent performance in a highly interactive workstation environment.

Material will also be drawn from the following software tools and algorithms:

- Random Number Generation
- Sorting
- Table look-up
- Mathematical functions

Some relevant aspects of DAP processing are:

- Several forms of internal communication
- Bit organisation
- Low control overheads when array processing
- Simultaneous high speed data display
- No local indexing
- Fortran-Plus

Some of the techniques for high DAP performance discussed:

Outer loops parallel, inner loops serial
Use of big memory embedded in a processor array
Whole problems/mapping problems top down
Oversize problems and data mappings
Recursive change of scale
Precision flexibility
Defining new higher level interfaces
Low level coding when appropriate
Activity control
Serial techniques
Parallelism analysis

The paper will draw together some of the author's experience of achieving high performance on DAP applications but it is not a general survey paper.

SOCIETY for INDUSTRIAL and APPLIED MATHEMATICS

1405 ARCHITECTS BUILDING • 117 SOUTH 17th STREET • PHILADELPHIA, PA 19103 • (215) 564-2929

INSTRUCTIONS

(For Contributed Papers and Poster Presentations)

1. Please use an elite typing element and carbon ribbon, if available.
2. Begin typing the abstract title, text, and author information directly below the heading captions. Type flush left, single-spaced within the space designated. Do not indent. (See enclosed sample.)
3. Abstract title should be underlined. Capitalize the first letter of all words, except articles, prepositions and conjunctions.
4. Abstract should not exceed 100 words. Please state the problems to be addressed, their relevance, the methodology and results. References, if necessary, should be in the body of the abstract. Formulas should be kept to a minimum—please, no vertical fractions, multiple subscripts, or handwritten symbols. Abstracts must be submitted on this form and will be printed as received. Errors in the text are the author's responsibility.
5. For two or more authors of the same affiliation and address, type the latter only once, directly below the authors' names.

IMPORTANT: ABSTRACTS MUST BE MAILED UNFOLDED.

PLEASE ANSWER THE FOLLOWING:

1. Your preference for presentation.
   Check one:
   (See reverse side for explanation)
   _____ Prefer Standard Presentation.
   ___✓___ Prefer Poster Presentation. Seitter

2. Visual Equipment
   Standard Presentation
   ___✓___ Overhead Projector
   One_____ Two_____
   _____ 2"x2" Slide Projector
   Other _____ None_____

   Poster Presentation
   ___✓___ Corkboard
   _____ Easel for Flip Chart
   Other _____ None_____

3. Subject Classification
   (Choose one or two from list on reverse side)
   33 41

ABSTRACT FORM

Please follow INSTRUCTIONS carefully before you complete the form below.

TITLE

Regeneration of Images from IFS codes on an Array Processor.

ABSTRACT

The Iterated Function System (IFS) codes of Barnsley and Sloan (Byte, Jan. 1988, p215) are based on fractal theory and permit high compression ratios on many natural images. A processor array such as the AMT DAP 510 can be used to regenerate such images at high speed. The algorithm used involves the Monte Carlo generation of many image points, sorting, compression to deal with duplicates, and then expansion onto an image space. An image involving one quarter of a million generated points can be created in less than a second.

Author(s) full name(s) and address(es) including department/division, city, state and zip code

S. F. Reddaway,
3 Woodforde Close, Ashwell,
Baldock, Herts. SG7 5QE England

Deadline for Submission of Abstract

(PLEASE COMPLETE PAGE 2)

SOCIETY for INDUSTRIAL and APPLIED MATHEMATICS

1405 ARCHITECTS BUILDING • 117 SOUTH 17th STREET • PHILADELPHIA, PA 19103 • (215) 564-2929

INSTRUCTIONS

(For Contributed Papers and Poster Presentations)

1. Please use an elite typing element and carbon ribbon, if available.
2. Begin typing the abstract title, text, and author information directly below the heading captions. Type flush left, single-spaced within the space designated. Do not indent. (See enclosed sample.)
3. Abstract title should be underlined. Capitalize the first letter of all words, except articles, prepositions and conjunctions.
4. Abstract should not exceed 100 words. Please state the problems to be addressed, their relevance, the methodology and results. References, if necessary, should be in the body of the abstract. Formulas should be kept to a minimum—please, no vertical fractions, multiple subscripts, or handwritten symbols. Abstracts must be submitted on this form and will be printed as received. Errors in the text are the author's responsibility.
5. For two or more authors of the same affiliation and address, type the latter only once, directly below the authors' names.

IMPORTANT: ABSTRACTS MUST BE MAILED UNFOLDED.

PLEASE ANSWER THE FOLLOWING:

1. Your preference for presentation.
   Check one:
   (See reverse side for explanation)
   _____ Prefer Standard Presentation.
   _____ Prefer Poster Presentation.

2. Visual Equipment
   Standard Presentation
   _____ Overhead Projector
   One_____ Two_____
   _____ 2"x2" Slide Projector
   Other _____ None_____

   Poster Presentation
   _____ Corkboard
   _____ Easel for Flip Chart
   Other _____ None_____

3. Subject Classification
   (Choose one or two from list on reverse side)
   _____ _____

ABSTRACT FORM

Please follow INSTRUCTIONS carefully before you complete the form below.

TITLE

Fast Evaluation of the Fractal Dimension of Boolean Images

ABSTRACT

Fast codes for data sets mapped as Boolean images have been implemented on the AMT DAP 510. by recursively dividing the image into boxes, the distance from the nearest member of the set is found for every pixel. A distance histogram is formed from which a fractal dimension is calculated. The Boolean work is especially fast on a bit-organised SIMD processor. Speed depends on the number of pixels, the dimensionality of the image space, whether the set size (N) is known, and the programming level. For 2**22 pixels, times range from 50 msecs for an ID Fortran-Plus code with unknown N, to 0.8 msecs for a 3D assembler code with N known.

Author(s) full name(s) and address(es) including department/division, city, state and zip code

Stewart F. Reddaway
3 Woodforde Close, Ashwell,
Baldock, Herts. SG7 5QE England

Deadline for Submission of Abstract

(PLEASE COMPLETE PAGE 2)

SOCIETY for INDUSTRIAL and APPLIED MATHEMATICS

1405 ARCHITECTS BUILDING • 117 SOUTH 17th STREET • PHILADELPHIA, PA 19103 • (215) 564-2929

INSTRUCTIONS

(For Contributed Papers and Poster Presentations)

1. Please use an elite typing element and carbon ribbon, if available.
2. Begin typing the abstract title, text, and author information directly below the heading captions. Type flush left, single-spaced within the space designated. Do not indent. (See enclosed sample.)
3. Abstract title should be underlined. Capitalize the first letter of all words, except articles, prepositions and conjunctions.
4. Abstract should not exceed 100 words. Please state the problems to be addressed, their relevance, the methodology and results. References, if necessary, should be in the body of the abstract. Formulas should be kept to a minimum—please, no vertical fractions, multiple subscripts, or handwritten symbols. Abstracts must be submitted on this form and will be printed as received. Errors in the text are the author's responsibility.
5. For two or more authors of the same affiliation and address, type the latter only once, directly below the authors' names.

IMPORTANT: ABSTRACTS MUST BE MAILED UNFOLDED.

PLEASE ANSWER THE FOLLOWING:

1. Your preference for presentation.
   Check one:
   (See reverse side for explanation)
   _____ Prefer Standard Presentation.
   _____ Prefer Poster Presentation.

2. Visual Equipment
   Standard Presentation
   _____ Overhead Projector
   One_____ Two_____
   _____ 2"x2" Slide Projector
   Other _____ None_____

   Poster Presentation
   _____ Corkboard
   _____ Easel for Flip Chart
   Other _____ None_____

3. Subject Classification
   (Choose one or two from list on reverse side)
   _____ _____

ABSTRACT FORM

Please follow INSTRUCTIONS carefully before you complete the form below.

TITLE

Fast Interactive Ising Codes on the AMT DAP 510

ABSTRACT

The traditional canonical algorithm, in which a fresh 24-bit random number is compared to the appropriate probability for every spin, has been implemented on the DAP 510 at 25 Mflips/sec with both magnetic field H and temperature T fully interactive and at 105 Mflips/sec with H=0. A micro-canonical code achieves 1250 Mflips/sec. A new algorithm creates 2 or 3 bit "demons" with controlled probabilities which are used many times over (with irregularity injected) to convert spin counts to flip decisions. For fully interactive codes, performance in Mflips or exchanges/sec is predicted at 230 for spin-exchange, 280 for spin-flip (400 with H=0). These performances are unmatched by any other machine. A video will be shown.

Author(s) full name(s) and address(es) including department/division, city, state and zip code

Stewart F. Reddaway
3 Woodforde Close, Ashwell,
Baldock, Herts. SG7 5QE England

Deadline for Submission of Abstract

(PLEASE COMPLETE PAGE 2)

THE DAP APPROACH

S F Reddaway

ICL
Stevenage
Herts
UK

Published in Infotech State of the Art Report
on Supercomputers Vol. 2 (1979)

© S F Reddaway 1979

S F REDDAWAY has worked for nearly nine years for ICL's Research and Advanced Development Centre at Stevenage, and prior to that for five years in London with English Electric Computers, a predecessor Company of ICL. In the early years he worked on engineering techniques associated with microelectronics, but gradually moved to working on computer architectures that could make the best use of advancing technology. This led to the DAP concept, on which he has been working for the last seven years, leading a group concerned with hardware, software and applications. Dr Reddaway graduated in Physics from Clare College, Cambridge University in 1962, and, from 1962 to 1965, was at Durham University from where he obtained his PhD in Applied Physics.

THE DAP APPROACH

HISTORY

The origin of the Distributed Array Processor (DAP) project lay in finding a match between large fairly regular number-crunching applications, such as meteorology, and a technology that could produce regular hardware very cheaply. It seemed that conventional architectures provided a very poor match, with complicated designs lacking in regularity and taking little advantage of the array properties of most large applications. Experience has shown that the range of applicability is much wider than the original target range.

As a research project, hardware, applications analysis and software were advanced together, resulting in a well integrated system. The project started in 1972; design of the pilot DAP (001) started in Autumn 1974, and it was operational by Spring 1976. The first application routines were written in assembly language, and a year later DAP-FORTRAN was in use. The pilot DAP, with a recently doubled store size, continues to be used for research.

This paper primarily describes a much larger first customer model that is currently being developed; there are some detailed changes from the pilot DAP, but the principles are the same. First delivery is scheduled for early 1980 to Queen Mary College, London University.

PRINCIPLES

The DAP comprises a few thousand processing elements (PEs) arranged in a two-dimensional array. The PEs are very simple but high processing power is achieved by having many of them. They execute a common instruction stream broadcast by a master control unit (MCU); the DAP is thus classified as a SIMD (Single Instruction Multiple Data) machine.

Various sizes of PE array can be made, offering a range of processing powers. It is most natural for the array to be a square whose side is simply related to standard store highway widths. The design aims for cost-effectiveness rather than speed at any price.

The PEs are bit-organised giving great flexibility. Hence parallelism can be exploited in operations such as table look-up, scanning data, image processing, symbol processing and sorting as well as arithmetic. Each PE has associated with it a few thousand bits of fast random access storage. The fast parallel transfer between stores and PEs balances the high processing speed. As well as being able to work in a bit-organised way on different data in each PE, word-organised processing is possible but with fewer data

items being processed in parallel.

The totality of PE stores form a standard store module of a conventional computer. Hence there is no need for separate transfer of data between host and DAP; the DAP does processing inside the store of the host and is under the control of the host via a simple interface. Being part of a general purpose system has two related advantages. It gives access to the facilities of that system, in particular the operating system, languages and input/output, and it allows users to progressively take advantage of DAP processing.

The first customer DAP has 4096 PEs arranged 64 × 64 each with 4K bits of fast store, making 2 Mbytes in all. There are 256 array boards, each containing the logic and storage for 16 PEs built from standard circuits. The use of LSI is covered in the next section.

Processing element

Figure 1 shows the essential features of one PE with some of the control and data paths omitted for clarity. All data paths are one bit wide.

The top multiplexor selects the input to the 'Arithmetic and Logic Unit' (ALU), which may be either the PEs own output or an output from its North, East, South, or West neighbour. The bottom multiplexor selects the PE output, which may be an ALU output, or store output, and may go to the store or to the MCU along row or column highways. Input data may also be broadcast from the MCU along row or column highways.

The ALU has three one-bit registers (A, Q and C) and a one-bit full adder. The low-level software allows total flexibility in the use of ALU facilities but most common usage is as follows.

Register A provides 'activity' control; certain store write operations are effective only if this register is true. This allows application of a function to selected elements of an array and is also used for bit-level implementation within functions. The A register incorporates a logical AND facility which allows conditions to be combined rapidly and may also be used in its own right for implementing general logic functions.

The Q register acts as a one bit accumulator and the C register as a carry store. The adder adds Q, C and the ALU input and its sum output may be written back to Q or the store and its carry output to C.

The carry out from the adder can also be routed to a neighbour, allowing an asynchronous 'ripply carry' along a line of PEs, which is used in the vector processing described later, and in applications such as image processing. The production DAPs reduce the number of connections by using the same paths for routing data to a neighbour.

A family of 'ADD TO STORE' instructions are performed in a read-modify-write mode, which is somewhat slower than most instructions, but they do reduce the inner loops of multiply from three to two instructions.

On production DAPs error checking includes PE logic as well as store by an ingenious scheme (006) that has parity PEs and a means of evaluating when parity is valid in the PE registers. When the DAP is acting as host storage a full Hamming scheme is maintained.

The DAP architecture is ideal for LSI, and the logic of four PEs, consisting of nearly

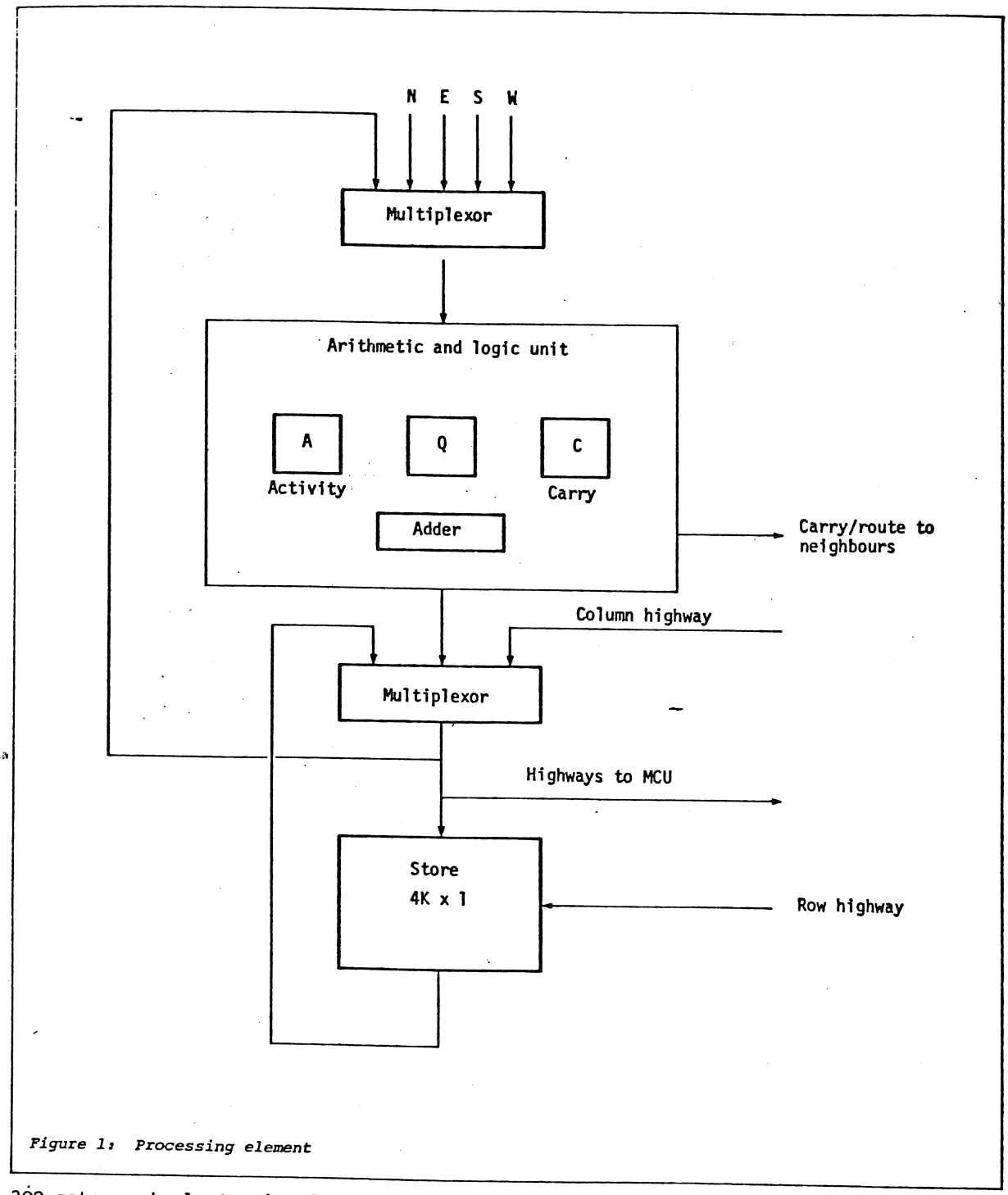


Figure 1: Processing element

200 gate equivalents, has been designed into a 28-pin circuit using an Uncommitted Logic Array (ULA) approach. Samples have been built into an array board which has 48 PEs (in place of 16) with extra PEs for byte parity. These circuits enable 128 × 128 DAPs to be built, and a design is far advanced which includes 16K bits/PE (32 Mbytes in all) and reconfigurability in the event of failure. LSI is also important for smaller DAPs in reducing the size, power consumption and construction cost, and in increasing the reliability.

Master control unit

Figure 2 is a schematic of the MCU, which may be likened to the instruction sequencing and control sections of a small computer with array instructions being implemented by

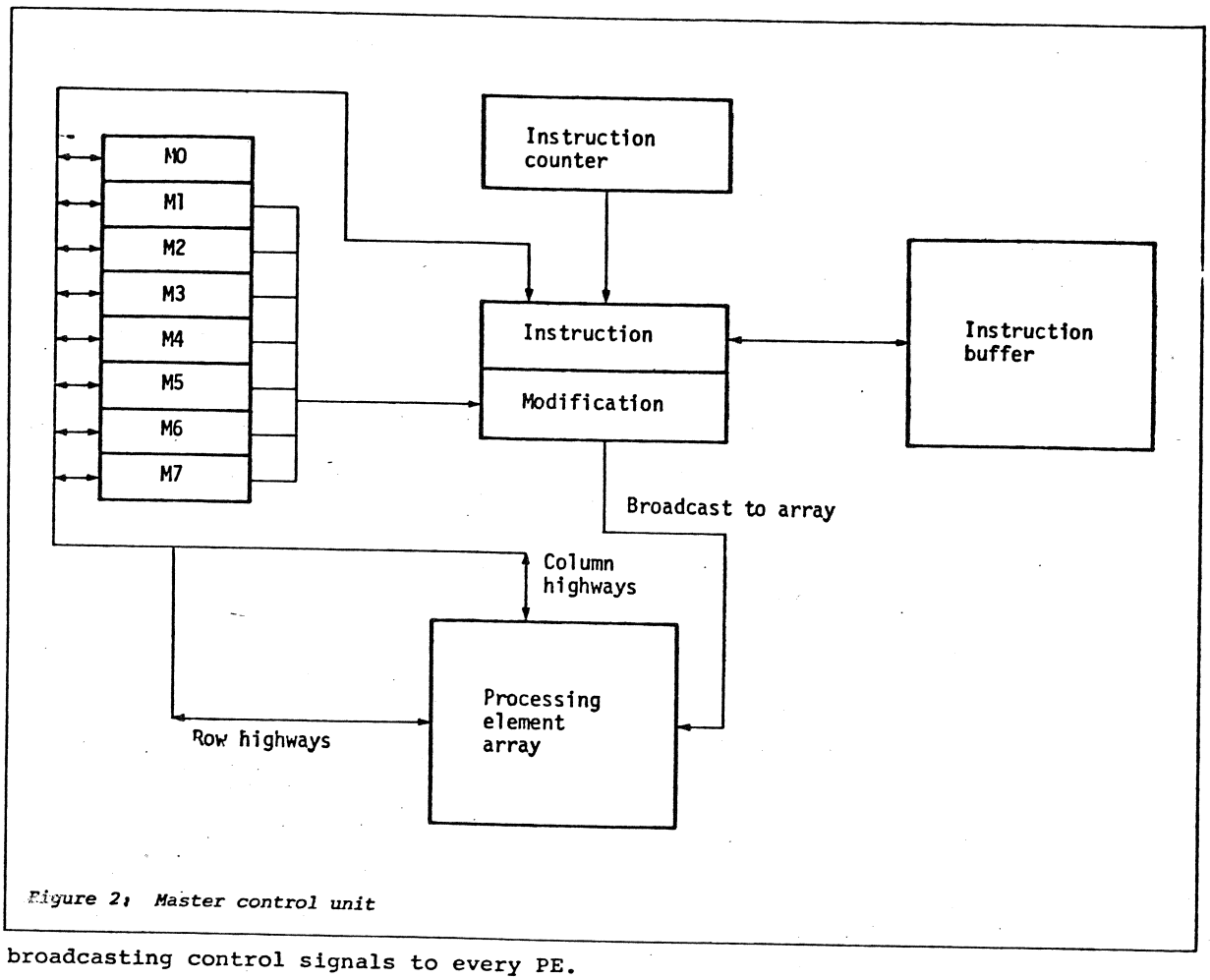


Figure 2: Master control unit

broadcasting control signals to every PE.

Instructions are fetched from the array store, each occupying 32 bits along a row of PEs. Successive instructions are spread over successive rows so that code is spread evenly among the PE stores. It must be emphasised that instructions can only be interpreted by the MCU; the PEs cannot recognise instructions themselves.

An important feature is the instruction buffer which stores loops (of up to 60 instructions) explicitly specified by a hardware instruction. Hence the instructions need to be fetched only once during the first time round the loop. Another important feature is the automatic stepping of addresses for operating on successive bits of a word.

There are eight general purpose registers, MØ to M7 with length equal to the side of the PE array. These may hold data, or addresses used in instruction execution. Scalar arithmetic can be done on these registers. The register contents may be transferred to or from the array along the row or column highways. This includes:

- Forming a matrix of bits in the array each of whose rows (or columns) equal the vector of bits from an MCU register
- Condensing a logical matrix in the array into a logical vector in an MCU register by logical ANDing along rows (or columns). If followed by a branch dependent on all elements of the MCU register being true, the latter can form part of a powerful 'global test'.

The normal control transfer instructions GOTO, LINK and EXIT and facilities for address arithmetic are provided.

The MCU also contains the store and control interfaces (not shown in Figure 2) to the host. When the DAP is processing, the host can still access the store by cycle stealing. Two consecutive rows of bits can be fetched at a time from the store to the MCU. This means that the 64 × 64 DAP can fetch four instructions in one cycle, or fetch 128 bits for 128-bit wide host store highways.

Modes of storage and processing

There are two formats in which data are held in the DAP store. In the 'vertical' format each number is held entirely within one PE with successive bits in successive store locations. In the 'horizontal' format each number is spread along a row of PEs, in a manner similar to the storage of DAP instructions; words as seen by the host are also in this format. The most powerful processing mode, known as matrix mode, operates on data in vertical format. Vector mode processing is used where problem parallelism is lower but makes less effective use of the PEs. This operates on data in horizontal format and makes use of ripple carry instructions. Processing on scalars can be done in either the MCU or the array. Simple integer operations can be done fast in the MCU registers. In the array, scalar operations are faster than vector, either because they can be made data dependent or because more parallelism can be used. An example of the latter is multiplication which uses a square of PEs of side equal to the mantissa length.

Transformations between horizontal and vertical format are done by DAP processing and take a time comparable with a multiply operation. Thus the overhead is normally negligible.

ARITHMETIC AND BASIC OPERATIONS

Since the PEs are bit-organised, arithmetic is built up by low-level software. Being software, it is always susceptible to improvement. Figure 3 gives times for some operations in standard ICL 2900/IBM 360 floating point format, based on a 200 nsec cycle time. Most of these are improvements on the times achieved with the pilot DAP (001). Unbiased rounding is used for greater accuracy than truncation towards zero. The operations are on matrices or vectors that match the DAP size.

Standard floating point has been implemented first because of its familiarity to users. However, for many applications a better solution is to use block floating point in which all elements of an array are normalised to a common (binary) exponent. Performance compared with standard floating point depends on the implementation and the application; typically, accuracy is at least as good, add/subtract is up to four times faster and multiplication time is comparable. It is likely to become important.

The times given in Figure 3 are for fully general routines that include store to store operation, activity control, normalisation, overflow checking, etc. This means that:

- It is possible to improve on these times by removing unnecessary generality and by merging the operations into each other
- There is little 'overhead' to be added to get program execution times.

For the operations in Figure 3 both operands and the result have the same rank. In

cases where the ranks are different the operation may be unexpectedly fast, as indicated by the examples in Figure 4.

Operation	Matrix	Vector	Scalar
Floating point (32 bit):			
Z←X+Y	150	38	20
Z←X*Y	250	45	27
Z←X/Y	330	90	
Z←X**2	125	35	
Z←SQRT(X)	170		
Z←X	17	1	
Z←MAX(X,Y)	33		
Z←LOG(X)	285		
Fixed point (32 bit):			
Z←X+Y	22	4	

Figure 3: Some approximate times (in μsec) for fully general routines. (Some times are slightly data dependent)

Operation	Time
Y←X*S	40-130
S←∑X	280
S←MAX(X)	48

Note: X and Y are matrices; S is a scalar.

Figure 4: Some times in μsec for 32-bit floating point mixed rank operations

Multiplication of a matrix by a scalar is a very frequent form of multiplication, and there are several cases to consider. Using a general routine, the average time for a random scalar is given by the top of the range given in Figure 4, and for 'simple' numbers, such as 3 or 0.25, the time is near the bottom of the range. It is also possible to write special routines for constants known at programming time (examples might be $\sqrt{2}$, Π, or 0.5) to gain substantial improvement. A special case is a power of 16 (or, if binary radix is used, two) which can be much faster than the fastest time shown.

Surprisingly, the sum of all elements of a matrix can be computed in less than twice the time to add two matrices. The method is described in (001).

A bit-level algorithm using 'global tests' is used to find the maximum element of an array.

An important activity can be scanning arrays of data for a match with a particular data item. This can be done at a rate of 1 bit/cycle/PE; or 2×10^{10} bits/sec for a 64 × 64 DAP. When a 'hit' is found the requirement may be to:

1 Do some processing on some associated data ('associative processing').
2 Retrieve some associated data.
3 Obtain the address of the hit(s).

For 3 and sometimes 2 binary chop procedures are useful.

The DAP offers complete generality and flexibility of function with very few decisions built into the hardware so the time taken by a function depends on its complexity. Some important consequences of this are:

1 The PEs are good not just at floating point arithmetic, but at Boolean processing, associative accessing and processing, image processing, sorting, symbol processing, fixed point processing – all with the same hardware. Of course problems must be in some sense array problems, but nearly all large problems are. The generality of the bit-organised array has been demonstrated by effective solutions being found for many requirements that were not considered when the hardware was designed.

2 The details of a function are left to the software. This means that the hardware does not dictate, for example, the functions available, the precision or the rounding algorithm.

3 Functions such as square root, logarithm and trigonometric functions are faster in comparison with basic arithmetic operations than on a conventional computer. This is because DAP can use various 'bit-level' algorithms that are unsuitable for the specialised hardware of conventional machines.

4 The DAP can take advantage of symmetries within a function so that, for example, the time for the squaring operation is about half that for general multiplication, and square root about half that for division.

5 Operations that are inherently simple may be several orders of magnitude faster than on conventional computers. For example, to replace every element of a matrix by its modulus takes less than 1 μsec for the whole matrix. Also, data dependent jumps in conventional programs are usually implemented on the DAP as conditional assignments. Thus the jumps are essentially replaced by setting the activity, which takes negligible time. Global tests, resulting in a branch if all elements of a Boolean matrix are true, takes less than 1 μsec.

6 For numerical work, most of the time is spent in the array arithmetic. This is in contrast to conventional machines which have special hardware to make selected arithmetic operations (for example, 64-bit floating point addition) very fast, which, apart from the benefit being confined to the precise functions selected, tends to make other operations the main bottleneck; examples might be:

- Conditional branches
- Reorganising data
- Store accesses
- Table look-ups
- Address arithmetic.

On DAP a large amount of arithmetic is done for the price of one set of organisational overheads.

There are two important consequences. The first is that the penalty of using a high-level language (DAP-FORTRAN) is small. The second is that the more (array) manipulative work that is involved, the better the relative performance of the DAP; the Mflops rate suffers less than that of other machines. Examples will be given later.

The complete flexibility of precision of data stored in vertical mode gives continuous space and speed tradeoffs. Arithmetic function times vary approximately linearly with precision, multiplication more sharply, addition less sharply.

SOFTWARE

For more details on software see (001) and (002). There follows a little on compiling programs and on the operating system.

DAP code translation

Two levels of programming are available: a macro assembly language and a parallel extension of FORTRAN known as DAP-FORTRAN. The DAP-FORTRAN compiler produces target code in a form suitable for input to the assembler: the assembly language itself, supplemented by a number of system macros.

After assembly the DAP subprograms, including any common blocks to be held in the DAP store, are consolidated into a contiguous block of code and data to be loaded into the DAP store. Entries from the host processor to specific DAP-FORTRAN subroutines within this block are engineered by 'interface procedures' which enable the subroutines to be called by the standard FORTRAN subroutine calling mechanism. These interface procedures are generated by the DAP-FORTRAN compilation system but are executed by the host.

Operating system software

The DAP makes minimal demands on the host operating system. The only special actions required are:

- The DAP program block is loaded into a contiguous area of the DAP store where it resides for the duration of the program (i e, it is non-paged and locked into store)
- The host treats sub-blocks of the program block as data areas corresponding to FORTRAN common blocks
- The host controls the DAP in a peripheral-like manner. This entails initiating processing by the DAP, after setting up store protection registers in the DAP, and servicing interrupts generated by the DAP when it stops processing.

PERFORMANCE ON SELECTED APPLICATIONS

A DAP system is a powerful and cost-effective tool for solving user problems, but it differs so greatly from other systems that comparisons are difficult. Most useful are comparisons of effectiveness on complete applications, and this includes the very important programming and diagnostic aspects not covered here. (The DAP-FORTRAN language and its use in various cases showing its elegance, conciseness and power is partly covered in (001,002,003,004 and 005). Performance on complete problems is more relevant than on particular algorithms with particular details, because for a given problem these may differ on different machines. Comparison at the level of arithmetic units is even more misleading.

All DAP performances reported below are for DAP-FORTRAN coding except for one in *Update on routines previously reported* and a few in *Update on applications previously reported* taken from (001); all, except for those in *Update on applications previously reported* and part of meteorological 'physics', are based on measured times on the pilot DAP. Figure 7 summarises most of the new performance figures reported below.

Some practical difficulties with performance comparison

Applications dependence

Due to the large spread in relative performance, figures are given only for particular applications.

Precision

Most machines restrict working to a very few precisions, sometimes only 64-bit for floating point; it is then difficult to make comparisons with more flexible machines, especially as the precision needed for particular variables is often not clear. The type of rounding used in the arithmetic can have a sizeable effect on the precision needed.

Difficulty of programming

Performance should be compared for roughly the same ease of programming. The best generalisation is probably to compare DAP-FORTRAN with FORTRAN, where one is roughly trading the inherently easier DAP-FORTRAN with its lesser familiarity. If assembly language is being compared, it is often useful to distinguish two levels on the DAP. The first uses the same standard arithmetic subroutines as DAP-FORTRAN and removes inefficiencies due to the high-level language; this level usually has only a small gain on the DAP. The second level is where everything is tailored to the problem; here considerably larger performance gains can often be made.

Algorithms

Often the optimum DAP algorithm differs from the optimum algorithm on other machines and this complicates comparisons.

Programming skill

Sometimes programs are not written as well as they could be, and this complicates comparisons between FORTRAN and DAP-FORTRAN codes.

Timing

Times on conventional machines have sometimes proved difficult to obtain and are often very compiler-dependent. Estimated times have sometimes had to be used for both the DAP and conventional machines. For numerical applications estimating times on the DAP is usually reliable, because most of the time is in the arithmetic.

Size of problem

This is covered in *Size effects* below.

Update on routines previously reported

Figure 5 differs from that in (001) in that the times are for larger problems on a 64 × 64 DAP, and, except for the convolution, are for DAP-FORTRAN (with 32-bit floating point). Details of the routines are given in (001).

Routine	Estimated time (msec)
Matrix multiply (64 × 64)	30
Matrix inversion with full pivoting (64 × 64)	52
FFT without reordering (4096 point complex)	15
Convolution, 16-bit integer (64 × 64)	5

Figure 5: Times for larger problems on a 64 × 64 DAP

The times for matrix multiply and FFT are about four times faster than on an IBM 360/195 or CDC 7600 with a good FORTRAN code and a good compiler. The matrix inversion is a more interesting case. The fastest FORTRAN time on the CDC 7600 that the author is aware of is 385 msec using the small core memory (SCM) (and 873 msec not using the SCM), giving a DAP improvement of 7.4 (or 16.8) times. This better performance compared with the other two routines is despite:

- An algorithm (Gauss-Jordan elimination) which performs some 50% more arithmetic than the Gaussian elimination used on sequential machines
- Full pivoting being used on DAP for better numerical stability than the usual partial column pivoting.

Why is DAP better on matrix inversion? In essence, the manipulative work associated with pivoting lowers the Mflops rate more on conventional machines than on DAP. Simple, regular arithmetic routines like matrix multiply tend to be ideal for showing off the floating point pipelines of conventional machines; actual problems tend not to be just simple and regular arithmetic.

The use of assembly language with the standard arithmetic subroutines rather than DAP-FORTRAN saves about 15% on matrix inversion, but only about 2% on matrix multiply. However, matrix multiply, for example, could be made about 30% faster by tailoring the arithmetic and manipulative work to the algorithm.

The convolution routine uses Fermat Number Transforms which exploit the DAP bit-level flexibility to achieve a very fast time.

Update on applications previously reported

The following table of performance estimates (Figure 6) is taken with minor amendment from (001).

More details are contained in (001). The estimated DAP time for the 'table lookup problem' has been nearly halved since (001) was written, because fast logarithm and exponential algorithms have been found, and faster table lookup techniques devised; however the relative performance in Figure 7 has not been changed because it is possible that on CRAY more use could be made of vector instructions than previously estimated.

Application	Estimated performance (64 × 64 DAP)
Finite element analysis	2-6 × 360/195
Simple relaxation	15 × 360/195
3-D magnetohydrodynamics	14-30 × 360/91
Many body simulation (galactic simulation)	10 × 7600
A data reorganisation problem	10 × 7600
A table lookup problem	3 × CRAY 1
A pattern matching problem	300 × 360/195
Operations research (The assignment problem)	1200 × 370/145

Figure 6: Performance estimates

Application	DAP FORTRAN	
	Time on 32 × 32 DAP	Relative performance on 64 × 64 DAP
Matrix inversion		7.4 or 16.8 × 7600
Meteorological dynamics		6.4 × 360/195
Meteorological 'physics'		11 × 360/195
Image processing		~350 × ICL 2970
Sorting 1024 32-bit items	7.6 msec	
Heat diffusion		200-460 × ICL 4-70
A many body problem		8 × 360/195
Hadamard transform on 2048 16-bit numbers	1.3 msec	
ADI (one iteration)	12.5 or 7 msec	

Figure 7: Relative DAP performances for various applications

A meteorological example

ICL have re-coded in DAP-FORTRAN a meteorological 'radiation physics' FORTRAN code known as HELIOS. This is highly data dependent; parts of the code depend on whether, in a particular vertical column of the atmosphere, clouds of various kinds exist and, if they do, at which of the 11 vertical layers and in what relationships to each other. It might be thought that, because parts of the code are only useful in a small number of vertical columns, the DAP performance (compared to conventional machines) is bound to be worse than on a less conditional code such as meteorological 'dynamics'. In fact the DAP is much better on HELIOS than on a 'dynamics' code, with both implemented in DAP-FORTRAN; the relative performance can be analysed into the following five factors (each in the range 1.25 to 1.6):

1 The obvious factor against the DAP is that many PEs are idle during conditional operations caused by differing cloud situations.

2 The corresponding factor for sequential machines is that tests and branches reduce the Mflops rate (compared with the regular 'dynamics'). This effect is bigger than that of 1.

3 There are many table lookup operations, and in the simple initial implementation, significant (and data dependent) time is taken by the DAP on these.

4 In analysing HELIOS, it became apparent that improvements could be made to the performance of the FORTRAN code; this is more significant than for 'dynamics'.

5 Logarithm is an important operation in HELIOS, and is very fast on DAP.

At the time of writing, the 'dynamics' code has been timed on the pilot DAP, but only on parts of HELIOS. Based on these timings, and reasonable estimates for the data dependencies, it is estimated that on a grid that fills a 64 × 64 DAP, the DAP is 6.4 times faster than an IBM 360/195 on the 'dynamics' and about 11 times faster than the existing FORTRAN on HELIOS, with all four codes using 32-bit floating point.

For large scale meteorology a 128 × 128 DAP with a large store is suitable. The large store may mean (the specification is not yet complete) a slower cycle time; if the machine is three times as powerful as a 64 × 64 DAP on a problem that fills it, the above performances become 19 and 33 times a 360/195.

For a grid that matches the DAP, HELIOS can fit comfortably in 4K bits/PE.

The DAP-FORTRAN is shorter and easier to read than the FORTRAN. For various reasons, the organisation of HELIOS for the two codes is significantly different, requiring effort to be spent in understanding HELIOS precisely. After that, about one-and-a-half man-months was spent in reprogramming the original 1000 lines of FORTRAN for the DAP. Programming a new meteorological model would take considerably less effort in DAP-FORTRAN than in FORTRAN.

Further speed improvements are possible. The 'dynamics' probably needs the 32-bit precision used (although accuracy is considerably improved on DAP by the unbiased rounding!), but radiation physics contains quite crude cloud parameterisations with low meteorological accuracy, making high precision meaningless. Also, meteorology lends itself well to block floating point. Finally, most of the time is spent performing table interpolations; this one activity can reasonably be coded in assembly language using improved table lookup methods. If block floating point is used throughout, with mantissa precision varying from 16 to 20 bits and table interpolation in assembler, HELIOS performance improves by a factor of about three.

Image processing

The performance figure in the table is for an application involving a smoothing function using 16-bit integers. The measured DAP performance was 30 times better than shown, but the reprogramming revealed a more parallel algorithm that could improve the original FORTRAN by about that amount. With 32-bit integers the measured time was 1.4 times slower, and with 32-bit floating point four times slower. (A 2970 is about 10 times slower than a CDC 7600.)

Other image processing work is discussed in (003); the scene analysis work has been implemented on the pilot DAP with very high performance.

A sorting example

A Batcher sorting algorithm has been implemented (002). A sort of 1024 32-bit items was implemented, but the code can readily be generalised for any quantity of data.

Heat diffusion

Three variants (with different boundary conditions) of a 2D heat problem and one 3D variant were implemented. Explicit finite difference schemes were used. The best performance was on the 3D problem. (An ICL 4-70 is some 30 times slower than an IBM 360/195.)

A many-body problem

A three-dimensional stellar evolution simulation in which each of 1024 stars interacts every time-step with every other star under gravitational force has been implemented. Such total interaction and the lack of a grid structure mean that it is not an obvious array processor problem. However, one and a half million force components are calculated each time-step in 3.3 seconds, giving a performance on this problem of about twice that of an IBM 360/195 on the pilot, or over 8 times on a 64 × 64 DAP.

Hadamard transform

A Hadamard transform, implemented with 2048 16-bit integers, is due to the use of low precision fixed point and the absence of multiplications, about 10 times faster than the 1024 point complex FFT dealt with in (001).

Alternating direction implicit (ADI) methods

ADI is mentioned in Section 3.1 of (003). The technique has been used on the pilot DAP to solve Poisson's equation on a 32 × 32 grid; a straightforward implementation took 150 msecs for 12 iterations, at which point it had nearly reached the final precision. An improved method does most of the computing on increments using shorter words and took 85 msecs for 12 iterations; at this point the error was rather less than the straightforward method and the final precision was substantially better. A 64 × 64 DAP will take slightly longer for 64 × 64 grids, but relative performance improves significantly on problems that are larger than the DAP size (003).

DISCUSSION OF PERFORMANCE

Maximum performance of 64 × 64 DAP

Particularly high performance (approximately 100 × CDC 7600) is obtained on applications such as image processing, the assignment problem, pattern matching or Hadamard transforms, where Boolean or low-precision integer processing is involved. Even larger figures (approximately 200-2000 × CDC 7600) can be obtained in particular instances, although without widespread significance; for example, inverting the sign of each element

of a matrix according to the sign of the corresponding element of another matrix. (A conventional machine would have to do three store accesses for each element, while DAP only needs two store accesses for a whole matrix of results.) This example might be part of a floating point calculation.

It is sometimes argued that performance on a routine-like matrix multiplication (approximately 4 × CDC 7600), in which the whole array is active should be the top performance for floating point. In terms of Mflops, DAP performs better on routines with less dependence on multiplication; but more important is the point, discussed below, that with conventional machines the Mflops rate on less straightforward routines often drops by more than on DAP. Examples with significantly better performance (approximately 10 × CDC 7600) are matrix inversion, the many-body problem, heat diffusion, the 'table lookup problem', and meteorological 'physics'.

As algorithms adapted to the DAP evolve, performance will rise further.

Factors that 'degrade performance'

It is sometimes argued that DAP performance will be considerably reduced by:

- Boundary effects
- Size effects
- Less parallel processing (such as implicit methods)
- Initial or final data manipulation
- Local data dependence ('inactive PEs')
- Data dependent flow of calculation (e g, pivoting) etc
- Scalar processing.

The general point is that most of the above effects usually cause *worse* degradation of the Mflops performance on conventional machines than on DAP. These effects are discussed below and in (003). Also discussed below are some other factors that often degrade performance of other machines such as CRAY compared with DAP.

Boundary effects

Often, by natural activity control, these can be incorporated into the 'interior' calculation. An example is the iterative solution of Poisson's equation, where an arbitrary boundary shape can be incorporated naturally. A hardware feature that can assist some cases is the geometry of the array edges: these can be 'plane' with zero shifted in at the edges, or 'cyclic' with rows (or columns) wrapping round on themselves to make loops. Both geometries can prove powerful, in terms of programming and performance. With three-dimensional problems, boundaries are usually surfaces, so that boundary processing can be programmed efficiently in matrix code. An example (not discussed in this paper) is transsonic aerodynamics.

Size effects

Clearly DAP performance will be significantly worse on a problem with, for example, arrays 67 × 67 than with 64 × 64. In most cases where a continuum is being simulated it is reasonable to adjust the overall scale (of two of the dimensions) to a power of two. Thus a 60 × 35 × 40 problem might become 64 × 32 × 40. Sometimes more complicated boundaries are used for continuum problems. An example from meteorology is a grid centred on the North Pole with an octagon outer boundary. DAP could map this onto a square and

not use all the PEs in the corners, or it could use all the PEs, with reduced merit being given to the computing in the corners. A slightly different meteorological approach uses a circular outer boundary that lies on the Equator, and this grid needs to be implemented exactly if the equatorial symmetry is used. The ease and efficiency of dealing with non-rectangular boundaries on the DAP often more than compensates for some PEs being wasted.

Non-continuum array problems, such as most finite element analysis, must usually take the problem size as given. However, the effect of mismatch to the DAP size is only important when dimensions are small or comparable with the DAP. Thus it is only a small factor with large finite-element problems.

Less parallel processing

Usually there is more parallelism than at first expected. Implicit methods are often held to have low parallelism; an example (ADI) is discussed above and in (003), and satisfactory performance is achieved.

Initial and final data manipulation

Sometimes an application has a large central arithmetic calculation, and also a substantial amount of largely manipulative work before and after; finite element analysis is an example. Study indicates that where this is substantial, then, to a greater or lesser extent, it contains a lot of parallelism that can be exploited effectively by DAP; often performance relative to other machines will be higher on this manipulative work than on the central arithmetic.

Local data dependence

On other machines tests and branches tend to disrupt the pipelines and slow down the Mflops rate, as well as being inelegant to program. There is also a tendency for a surprisingly large amount of unconditional work to be needed to determine whether a conditional computation should be done. Local data dependence is important in the meteorological 'physics' example discussed above.

Data dependent flow of calculation

The example of matrix inversion with pivoting was discussed above.

Memory bottleneck on other machines

On conventional machines the transfer rate in and out of memory is usually a bigger bottleneck than the arithmetic rate, but on DAP arithmetic times are memory-to-memory.

Inflexible precision

If, for example, either 40 or 80-bit precision is required when only 64-bit is available, then inefficiency results not just in the processing, but also in the memory space and input/output requirements.

Range of functions

As mentioned above, many functions haye high performance on DAP. On a conventional machine these would largely be implemented with addition and multiplication (for example,

a power series for logarithm), and this illustrates another way that raw Mflops comparisons are misleading.

Scalar performance and scalar processing

There is a school of thought that says that because significant amounts of a calculation on a vector machine cannot make use of vector instructions, the 'scalar performance' is liable to dominate; criticism is then made of the not very powerful scalar floating point arithmetic in either the DAP or the host.

Our experience is that significant sized applications tend to have a high degree of parallelism exploitable by the DAP with DAP-FORTRAN. Three reasons for this apparent contradiction are discussed below. However, a 'scalar' processing may be important during a transition period if old large codes need to be run only partly converted to DAP-FORTRAN.

1 Much of the inherent parallelism of applications is disguised when writing in a sequential language such as FORTRAN. Therefore it is not surprising if either compilers or human code converters have difficulty reconstructing the parallelism. But with a well constructed parallel language such as DAP-FORTRAN, it is easier to program the originally parallel application, and much greater parallelism results.

2 Many vector processors are limited in their flexibility and range of parallel functions, particularly when they are programmed from a high-level language. An important example is conditional operations where ordinary FORTRAN will contain branches and an automatic compiler will usually fail to vectorise, even if the vector machine contains some relevant facilities. Other examples are various kinds of table lookup, associative processing and sometimes operations like finding a maximum element or the sum of an array. Experience on such machines has little relevance to DAP.

3 Different algorithms can often increase the fraction of parallel or vector work.

The conclusion from 1 and 2 above is that the importance of scalar facilities in vector machines is more a reflection on those systems than on the nature of applications. It might also be remarked that after considerable difficulty some large codes are now running on CDC STAR with 99.9% 'vectorisation'.

Concern is also sometimes expressed about performance on short vectors, as this seems to be important on vector machines. The above three reasons are also applicable here, and in addition:

- A vector machine system is oriented towards one dimension, and hence encourages short vectors
- DAP vector arithmetic deals naturally with most one-dimensional boundary processing
- Experience shows that much less DAP time is spent on vector arithmetic than on matrix arithmetic.

Range of applicability

A wide range of applications (most CPU bound applications seem to be included) can be

done effectively on DAP. 'Effectiveness' can only be measured against other systems, and naturally performance varies. If in practice DAP is more cost-effective on the majority of applications, then in some sense conventional machines are 'special purpose'! In several respects they are more restricted having:

- A specific word length
- Operations on only one pair of words at a time
- No effective associative accessing or processing
- Inefficient Boolean processing
- Specialised hardware limited to selected functions.

DAP is restricted in the sense that it currently uses an adaptation of an existing language. This must be weighted against the ease of DAP programming and cost-effective execution.

DAP is new and different, so there are certain to be many exploitation techniques awaiting discovery, particularly on less well studied applications. Research is going on in several areas, including compilation and such commercial DP activities as payroll and sorting. Operating systems are one of many areas that seem well worth investigating.

Parallel programming

Parallel languages are more natural for problems which are inherently parallel; if an application is broadly suited to the DAP, DAP-FORTRAN programming is usually easier than FORTRAN. Indeed, current proposals for array extensions to FORTRAN have many features similar to DAP-FORTRAN. While DAP-FORTRAN is easy for a FORTRAN programmer to learn, it is probably true that a sequential programmer must 'unlearn' much of his way of thinking; a new programmer will probably take to parallel programming more easily.

Algorithms

This is partly covered in (003). Most applications can be tackled effectively with, essentially, the existing algorithms. However, new or different algorithms may be better than existing ones, and the improvement may be such that the DAP would not be effective on that application without it.

The degree of algorithmic difference can range from rearranging the calculation without changing the mathematics, up to radically new algorithms. Matrix multiplication is an example of the former, whereas matrix inversion involves changed mathematics. The start of some iterative methods is slightly different mathematically. An example of major change would be a move from an implicit to an explicit method.

Because of the possible benefits of rearranging or changing algorithms, it is best to approach applications top-down for DAP implementation.

COMMENTS ON THE ARCHITECTURE

SIMD or MIMD?

MIMD loses the advantage of simplicity, particularly if it attempts to have many processors that can either co-operate on large jobs or be split on many jobs. Programming and control seem impossibly complex and inefficient. Cost-effectiveness is lost both by the extra control hardware, and by wasted execution time. Even if these problems were surmounted, the generality and flexibility gained by multiple instructions must be weighed against that lost if bit organisation is given up. MIMD may provide interesting work for computer scientists, but not for cost-effective computing, especially for a wide range of large jobs. More limited MIMD may have some success.

Bit organisation

The essential advantages are the simplicity, generality and flexibility previously mentioned. Simplicity has many hardware advantages, especially in an era of rapid technology advance; generality and flexibility have many user advantages that will be increasingly exploited.

Compared with, say, 4- or 8-bit designs, a 1-bit design loses nothing in performance per gate. Compared with more specialised word-organised hardware, bit-organisation may lose a little in terms of performance per gate for the artificial requirement of doing continuous 64-bit multiplication, but all the simple, general and flexible advantages remain, and word machines have other bottlenecks such as memory highways and control overheads.

Two-dimensional connectivity

In essence, connectivity is expensive in hardware terms, but lack of connectivity is expensive in terms of time spent on routing. With 1-D, operations such as FFT are dominated by routing, but not with 2-D; 3-D is expensive in terms of connections. Also, 2-D connectivity maps naturally onto the construction technology. Having many neighbours (for example diagonal neighbours) adds little to performance.

COMMENTS ON THE FUTURE

1 The demonstrated range of applicability of DAP-like machines will expand, including non-scientific processing.

2 Some of the factors affecting the rate of expansion are:

- The extent to which users and purchasers grasp and exploit the improved problem solving opportunities offered by DAP
- The rate at which DAP-like hardware and software products are developed and marketed, and the pricing policies followed. A hypothetical product might be a low-priced 32 × 32 DAP engineered for volume production and connectable to a mini-computer (or else stand-alone). Another example is a DAP-FORTRAN that includes vectors and matrices of any size

- The rate at which library and applications packages are produced is important.

3 As applications broaden, one line of development may be towards 'active store' that is not much more expensive than ordinary store. The culmination may eventually be that computer manufacturers cease to produce ordinary store.

CONCLUSIONS

DAP is a radically new approach that is simple, general and flexible, and will effectively exploit advancing LSI technology. Performance is high and cost-effective across a wide range of applications, and programming is easy with a powerful language, DAP-FORTRAN. There is plenty of scope for further development.

REFERENCES

001 FLANDERS P M, HUNT D J, REDDAWAY S F and PARKINSON D
Efficient high-speed computing with the distributed array processor
High-speed Computer and Algorithm Organisation pp 113-128
Academic Press Inc (1977)

002 FLANDERS P M
FORTRAN extensions for a highly parallel processor
In *Supercomputers* Infotech State of the Art Report Infotech Intl Ltd Maidenhead Berks (1979)

003 HUNT D J
Application techniques for parallel hardware
In *Supercomputers* Infotech State of the Art Report Infotech Intl Ltd Maidenhead Berks (1979)

004 GOSTICK R W
Software and algorithms for the distributed array processor
ICL Tech J vol 1 issue 2
(May 1979)

005 GOSTICK R W
Supercomputer diagnostics
In *Supercomputers* Infotech State of the Art Report Infotech Intl Ltd Maidenhead Berks (1979)

006 HUNT D J
UK Patent Appl no 45858/78

From Electronics to Microelectronics
W.A. Kaiser and W.E. Proebster (eds.)
© North-Holland Publishing Company, 1980

REVOLUTIONARY ARRAY PROCESSORS

Dr. S.F. REDDAWAY
International Computers Limited,
Research and Advanced Development Centre,
Fairview Road, Stevenage, Herts, England.

In a world full of parallelism, distributing an array of simple and flexible processing elements throughout a main memory module can both provide natural applications programming and exploit microelectronics effectively.

This paper discusses a style of array processing exemplified by the Distributed Array Processor (DAP). More details on the design, programming, applications and performance are given in the references.

1. COMPUTER HISTORY

Early computers were designed as a natural progression from manual methods of processing, which, especially for mathematical calculations, were sequential. Problems may have been thought about in parallel terms - for example, selecting all records that satisfy a certain criteria, calculating interest charges on all bank accounts, or multiplying two matrices together - but the actual work was reduced to sequential steps. Early electronic computers were designed for mathematical use, and considerable resource was devoted to operations on pairs of numbers; instruction sets evolved accordingly. As technology developed, it was natural to make the individual operations faster and the flow of control more efficient rather than making more radical changes. In the search for power, special hardware was built for some operations, but they remained confined to pairs of numbers. The reasons for this line of development seem to have been partly lack of imagination on the part of designers and partly that users were brainwashed by early machines (and earlier manual methods) into thinking that the art of programming consisted of reducing problems to sequential form. This, together with the sequential programming languages that evolved, led to the wrong conclusion that most problems are essentially sequential.

2. DAP BACKGROUND

The above trends tended to reinforce each other, and taking a narrow viewpoint there seemed to be powerful arguments against radical change. Several lines of thought that have contributed to DAP evolution are discussed below. Taken separately they would have little impact, but the team that originated the DAP covered hardware, software and applications, and by taking a total view created a radical system with revolutionary potential.

The DAP itself is briefly described in section 3.

(a) Most problems have a high degree of parallelism. The natural parallelism, which can take many forms, is often lost when applications are coded in sequential languages, and the extent of parallelism in applications is often underestimated. In particular, a lot of processing can be viewed as operating on arrays of data, something that the DAP does directly.

(b) Development of Parallel Algorithms. As well as sequential languages obscuring parallelism, most algorithm development has been for sequential machines. Often improved parallel algorithms can be found. Research in this area has been hampered by both a lack of parallel languages and a lack of understanding of what is offered by hardware such as DAP.

(c) Parallel Languages. As indicated above, the inertia of sequential languages is very great, and considerable effort has gone into compiling sequential languages for some parallel machines. This approach is mostly disappointing in terms of performance, does not encourage algorithm development, and does not achieve the ease-of-use gains possible with a good parallel language.

Other parallel languages have been developed, notably APL. The main interest in APL has been the mathematical and logical clarity that can be achieved by parallel structures, rather than efficient hardware execution or its application to large problems.

The DAP approach is very different, providing for a parallel world a parallel language that executes efficiently on parallel hardware; the improvement in terms of effectiveness in

solving problems is very marked.

(d) Falling hardware costs. DAP exploits these without running into either:

(1) physical limitations (due, for example, to the speed of light) experienced by conventional machines;

or

(2) complexities and falling effectiveness experienced by multiple processor approaches that rely on the concurrent execution of different pieces of sequential code.

(e) Two significant hardware technology developments have been

(1) that it is much cheaper to produce many copies of simple designs than few copies of large complex designs,

and

(2) that memory and logic are now made from the same technology.

The DAP exploits both these by large-scale replication of very simple Processing Elements (PE) and by physically mixing storage and processing.

(f) Functions from bit-level operations. The very simple PEs resulting from (e) are bit-organised. This means they are very general, in that all kinds of function can be built up by software, and very flexible in that the details are determined by software rather than hardware. The fact that everything is ultimately derived from the same set of bit-level operations leads to:

(1) a unified view in computer science terms which often gives fundamental insights into functions and algorithms;

and

(2) all functions using the same hardware, so that (provided operations are on arrays of data) it is continuously active. This contrasts with specialised hardware, such as a multiplier, which is likely to have low utilisation. This implies that one should beware of hardware put in to help a particular algorithm; the algorithm may change, and in any event the hardware is unlikely to be cost-effective overall.

Research into the use of bit-organised PEs and arrays has not been extensive in the past, but the impetus given to it by the DAP is yielding many effective and surprising results. The lesson seems to be "keep it simple, keep it general, keep it flexible".

(g) The place of DAP in a complete system. The DAP is closely integrated into a conventional system giving efficient use of existing facilities and software.

3. DESCRIPTION OF THE DAP

More detailed descriptions are contained in [1], [4] and [5]. The DAP comprises a few thousand processing elements (PEs) arranged in a two-dimensional array. The PEs are very simple but high processing power is achieved by having many of them. They execute a common instruction stream broadcast by a master control unit (MCU); the DAP is thus classified as a SIMD (Single Instruction Multiple Data) machine.

Various sizes of PE array can be made, offering a range of processing powers. It is most natural for the array to be a square whose side is simply related to standard store highway widths. The design aims for cost-effectiveness rather than speed at any price.

The PEs are bit-organised (i.e. all data paths are one bit wide) giving great flexibility. Hence parallelism can be exploited in operations such as table look-up, scanning data, image processing, symbol processing and sorting as well as arithmetic. Each PE has associated with it a few thousand bits of fast random access storage. The fast parallel transfer between stores and PEs balances the high processing speed. As well as being able to work in a bit-organised way on different data in each PE, word-organised processing is possible but with fewer data items being processed in parallel.

The totality of PE stores form a standard store module of a conventional computer. Hence there is no need for separate transfer of data between host and DAP; the DAP does processing inside the store of the host and is under the control of the host via a simple interface. Being part of a general purpose system has two related advantages. It gives access to the facilities of that system, in particular the operating system, languages and input/output, and it allows users to progressively take advantage of DAP processing.

The first customer DAP has 4096 PEs arranged 64 x 64 each with 4K bits of fast store, making 2 Mbytes in all. There are 256 array boards, each containing the logic and storage for 16 PEs built from standard circuits.

The first such machine is already working on ICL premises, and the second will be delivered to Queen Mary College, London University in early 1980. A 32 x 32 pilot model DAP has been working since 1976.

There are two features that make the array two-dimensional; row and column highways connect all the PEs in a row or column, and each PE is connected to its four nearest neighbours.

The PE contains three one-bit registers, one of which provides 'activity' control; certain store write operations are effective only if this register is true. This allows application of a function to selected elements of an

array and is also used for bit-level implementation within functions.

The DAP architecture is ideal for LSI, and the logic of four PEs, consisting of nearly 200 gate equivalents, has been designed into a 28-pin circuit using a Uncommitted Logic Array (ULA) approach. Samples have been built into an array board which has 48 PEs (in place of 16) with extra PEs for byte parity. These circuits enable 128 x 128 DAPs to be built, and a design is far advanced which includes 16K bits/PE (32 Mbytes in all) and reconfigurability in the event of failure. LSI is also important for smaller DAPs in reducing the size, power consumption and construction cost, and in increasing the reliability.

The MCU may be likened to the instruction sequencing and control sections of a small computer with array instructions being implemented by broadcasting control signals to every PE.

There are two formats in which data is held in the DAP store. In the 'vertical' format each number is held entirely within one PE with successive bits in successive store locations. In the 'horizontal' format each number is spread along a row of PEs; words as seen by the host are also in this format. The most powerful processing mode, known as matrix mode, operates on data in vertical format. Vector mode processing is used where problem parallelism is lower but makes less effective use of the PEs; this operates on data in horizontal format. Processing on scalars can be done in either the MCU or the array.

Transformations between horizontal and vertical format are done by DAP processing and take a time comparable with a multiply operation. Thus the overhead is normally negligible.

Since the PEs are bit-organised, arithmetic is built up by low-level software. For example, with 32-bit matrices, integer addition takes about 100 cycles and floating point addition about 800 cycles; a cycle is about 200 nsec.

4. SOME CONSEQUENCES OF THE DAP'S BIT-ORGANISATION

The DAP offers complete generality and flexibility of function with very few decisions built into the hardware, so the time taken by a function depends on its complexity. Some important consequences of this are:

(1) The PEs are good not just at floating point arithmetic, but at Boolean processing, associative accessing and processing, image processing, sorting, symbol processing, fixed point processing - all with the same hardware. Of course problems must be in some sense array problems, but nearly all large problems are. The generality of the bit-organised array has been demonstrated by effective solutions being found for many requirements that were not considered when the hardware was designed.

(2) The details of a function are left to the software. This means that the hardware does not dictate, for example, the functions available, the precision or the rounding algorithm.

(3) Functions such as square root, logarithm and trigonometric functions are faster in comparison with basic arithmetic operations than on a conventional computer. This is because DAP can use various 'bit-level' algorithms that are unsuitable for the specialised hardware of conventional machines.

(4) The DAP can take advantage of symmetries within a function so that, for example, the time for squaring is about half that for general multiplication, and square root about half that for division.

(5) Operations that are inherently simple may be several orders of magnitude faster than on conventional computers. For example, to replace every element of a matrix by its modulus takes less than 1 μsec for the whole matrix. Global tests, resulting in a branch if all elements of a Boolean matrix are true, take less than 1 μsec.

(6) For numerical work, most of the time is spent in the array arithmetic. This is in contrast to conventional machines which have special hardware to make selected arithmetic operations (for example, 64-bit floating point addition) very fast, which, apart from the benefit being confined to the precise functions selected, tends to make other operations the main bottleneck; examples might be:

* Conditional branches
* Reorganising data
* Store accesses
* Table look-ups
* Address arithmetic

On DAP a large amount of arithmetic is done for the price of one set of organisational overheads.

There are two important consequences. The first is that the penalty of using a high-level language (DAP-FORTRAN) is small. The second is that the more (array) manipulative work that is involved, the better the relative performance of the DAP; the Mflops rate suffers less than that of other machines.

The complete flexibility of precision of data stored in vertical mode gives continuous space and speed trade-offs. Arithmetic function times vary approximately linearly with precision, multiplication more sharply, addition less sharply.

5. SOFTWARE

Most programming is done in a new language developed for the DAP known as DAP-FORTRAN [2]. Its facilities are tailored to the type of operation needed for processing arrays, but there is a close match to the hardware

structure, thus ensuring effective use of the machine.

DAP-FORTRAN is basically an array processing extension of FORTRAN. It is aimed mainly at numeric applications for a highly parallel array processor, and aims to encourage efficient and natural exploitation of parallelism.

The position of DAP as part of a general purpose system is reflected in the overall program structure; total processing is split between host and DAP in a way which is appropriate to the job, with all input-output being performed by the host. Code to execute on the host is programmed in standard FORTRAN with calls to DAP-FORTRAN subroutines which are executed by the DAP. Processing within the DAP is initiated and monitored by the host in much the same way as a peripheral transfer.

Partitioning of DAP and host processing at the subroutine level reflects the loose coupling between the two and is consistent with the overheads expected in the host for process switching in a multi-programming environment. Since the DAP is an integral part of the host's store, data communication between the two parts can be achieved via common data areas situated in the DAP store, without any movement of data. Any necessary storage mode conversions are performed by the DAP either explicitly by calls to system subroutines or implicitly during execution of DAP-FORTRAN statements.

6. PERFORMANCE

Reference [4] discusses performance in general, and includes applications such as matrix operations, FFT, convolutions, meteorology, image processing, sorting, heat diffusion, many body problems and alternating direction implicit (ADI) methods.

A DAP system is a powerful and cost-effective tool for solving user problems, but it differs so greatly from other systems that comparisons are difficult. Most useful are comparisons of effectiveness on complete applications, and this includes the very important programming and diagnostic aspects not covered here. (The DAP-FORTRAN language and its use in various cases showing its elegance, conciseness and power is partly covered in ([1], [2], [3], [5], [6] and [7]). Performance on complete problems is more relevant than on particular algorithms with particular details, because for a given problem these may differ on different machines. Comparison at the level of arithmetic units is even more misleading.

On a range of floating point routines and applications, performance of a 64 x 64 DAP is typically between 4 and 15 times a machine like CDC 7600, with the higher figures usually associated with less straigtforward routines. Particularly high performance (approximately 100 x CDC 7600) is obtained on applications such as image processing, the assignment problem, pattern matching or Hadamard transforms, where Boolean or low-precision integer processing is involved. Even larger figures (approximately 200-2000 x CDC 7600) can be obtained for specific operations.

7. RANGE OF APPLICABILITY

A wide range of applications, which seems to include most CPU bound applications, can be done effectively on DAP. In several respects conventional systems are more restricted:

(1) a specific word length;
(2) operations on only one pair of words at a time;
(3) no effective associative accessing or processing;
(4) inefficient Boolean processing;
(5) specialised hardware limited to selected functions.

DAP is new and different, so there are certain to be many exploitation techniques awaiting discovery, particularly on less well studied applications. Research is going on in several areas, including compilation and such commercial d.p. activities as payroll and sorting. Operating systems are one of many areas that seem well worth investigating.

8. PARALLEL PROGRAMMING

Parallel languages are more natural for problems which are inherently parallel; if an application is broadly suited to the DAP, DAP-FORTRAN programming is usually easier than FORTRAN. Indeed, current proposals for array extensions to FORTRAN have many features similar to DAP-FORTRAN. While DAP-FORTRAN is easy for a FORTRAN programmer to learn, it is probably true that a sequential programmer must un-learn much of his way of thinking; a new programmer will probably take to parallel programming more easily.

Data dependent jumps in sequential code are normally replaced by masked assignments in DAP-FORTRAN. These use the hardware activity control, which is very fast compared with jumps on conventional machines. Although PEs that are de-activated may be said to be **idle**, it is remarkable in highly conditional codes how much time conventional machines spend performing tests and jumps rather than the conditional operations themselves. Thus DAP often performs best on this type of code, even if for part of the time very few PEs are active. Although global tests and branches can be used, DAP code is often a superset (without branching) of the code required for all possibilities. It is then important to minimise the code that is different for the various conditions, and experience shows that commonality is often much greater than is apparent from a sequential coding; greater clarity and insight often results from the DAP-FORTRAN coding.

Most applications can be tackled effectively with essentially the existing algorithms. However, new or different algorithms may be better than existing ones.

Because of the possible benefits of re-arranging or changing algorithms, it is best to approach applications top down for DAP implementation.

9. COMMENTS ON THE FUTURE

The demonstrated range of applicability of DAP-like machines will expand, including non-scientific processing. Some of the factors affecting the rate of expansion are:

* the extent to which users and purchasers grasp and exploit the improved problem solving opportunities offered by DAP
* the rate at which DAP-like hardware and software products are developed and marketed, and the pricing policies followed. A hypothetical product might be a low-priced 32 x 32 DAP engineered for volume production and connectable to a mini-computer (or else stand-alone)
* the rate at which library and applications packages are produced.

As applications broaden, one line of development may be towards 'active store' that is not much more expensive than ordinary store. The culmination may eventually be that computer manufacturers cease to produce ordinary store.

REFERENCES

[1] P.M. Flanders, D.J. Hunt, S.F. Reddaway, and D. Parkinson, Efficient high-speed computing with the Distributed Array Processor, in High speed Computer and Algorithm Organisation pp 113 - 128. (Academic Press Inc. 1977).

[2] P.M. Flanders, FORTRAN extensions for a highly parallel processor, in Supercomputers, Infotech State of the Art Report (Infotech Intl Ltd Maidenhead Berks 1979) 117-133.

[3] D.J. Hunt, Application techniques for parallel hardware, in Supercomputers, Infotech State of the Art Report (Infotech Intl Ltd Maidenhead Berks 1979) 205-219.

[4] S.F. Reddaway, The DAP Approach, in Supercomputers, Infotech State of the Art Report (Infotech Intl Ltd Maidenhead Berks 1979) 309-329.

[5] S.F. Reddaway, D.J. Hunt, P.M. Flanders, Application of the ICL DAP. Convention Informatique, Paris (Sept 1979)

[6] R.W. Gostick, Software and algorithms for the Distributed Array Processor, ICL Tech J. Vol 1 issue 2. (May 1979)

[7] R.W. Gostick, Supercomputer diagnostics, in Supercomputers, Infotech State of the Art Report (Infotech Intl Maidenhead Berks 1979) 135-146.

Stewart Reddaway is Manager of Parallel Processing at ICL's Research and Advanced Development Centre in Stevenage, England. He received his BA degree in physics from Clare College, Cambridge University, in 1962 and got his Ph.D. in Applied Physics from Durham University, where he was from 1962-1965. Dr. Reddaway has worked for 14 years in research with ICL and English Electric Computers, a predecessor company of ICL. In the early years he worked on engineering techniques associated with microelectronics, but gradually moved to working on computer architectures that could make the best use of advancing technology. This led to the DAP concept, on which he has been working for the last seven years, leading a group concerned with hardware, software and applications.

Workshop on "~~Computer~~ Parallel Architectures and Computer Vision"
Oxford, March 1987

MAPPING IMAGES ONTO PROCESSOR ARRAY HARDWARE

Stewart Reddaway, ICL

1. INTRODUCTION

Vision images typically contain 10^4 to 10^6 pixels. Whilst some architectures aim to achieve high performance with serial techniques using pipelining and dedicated hardware, highly parallel processor arrays are becoming increasingly attractive. For increasingly complex vision processing, it is advantageous:

(a) to mix memory and processing in a processor array so as not to overload a separate memory,

and

(b) for such an array to be highly programmable.

A generally appropriate degree of communication is usually provided by a 2D structure, and this can be particularly appropriate for image processing.

A processor array machine familiar to the author is the DAP. ICL developed the DAP-2 (also known as MiniDAP or MilDAP) and several prototypes have been delivered; DAP-3 is a re-engineered faster machine with more memory that is being produced by a new company, AMT. The DAP architecture has been described in many papers, for example in Reddaway (1987). Briefly, it is an SIMD 2D array of simple Processing Elements (PEs); it has nearest neighbour connections with cyclic (or plane) boundaries. DAP-2 and DAP-3 have 1024 PEs arranged 32 x 32, and each PE has tens of thousands of bits of memory.

Despite having 1024 PEs, DAP and similar machines have many fewer PEs than there are pixels in a typical image; hence consideration must be given as to how these "oversize" images are mapped into the DAP memory which, collectively, is big enough in nearly all cases. The aim of this paper is to set out some of the ways image data can be mapped onto processor arrays, and some of the advantages and disadvantages for various algorithms.

2. MAPPINGS FOR 2D PROCESSOR ARRAYS

There are 2 broad approaches, known as sheet and crinkled, to oversize mappings; it is also possible to mix them.

2.1 Sheet Mapping

The pixel image is divided into processor array sized sheets (32 x 32 on DAP-2 and DAP-3) and each sheet is mapped across the processor array. As an example, a 512 x 512 image will have 256 sheets which can be organised and indexed as a 16 x 16 data structure. An individual PE contains pixels sampled uniformly across the whole image in both dimensions, with a sampling interval of 32 pixels.

Processing such a structure requires special care and extra processing operations to cater for the sheet boundaries. Also, nearest neighbour pixels are always in a neighbouring PE, and hence extra operations are always inolved in accessing a neighbouring pixel. (More distant pixels are correspondingly further away in PE terms, until linear distances of more than 16 are involved; then cyclic geometry means that PE distance reduces until the 32nd neighbour pixel is in the same PE).

If processing is needed only on a limited region of the image, then this mapping allows the full power of the processor array to be used. A relevant example might be "blob extraction".

Address calculation overheads can often be reduced by copying the current sheet from the main indexed data structure to a work area. Nevertheless, control overheads can still be significant in dealing with sheet boundaries.

2.2 Crinkled Mapping

The image is divided into as many local areas as there are PEs, and each PE contains one such local area. As an example, a 512 x 512 image crinkle mapped onto a DAP would be divided into 1024 local areas of 16 x 16 pixels. These local areas are assigned one to a PE in a regular way across the processor array. A DAP matrix contains pixel samples uniformly spaced across the whole image; for the example, the sampling interval is 16 pixels in both dimensions. The crinkled name derives from squashing (and hence crinkling) the image until it fits the processor array dimensions. The term pyramid mapping has also been used.

Processing whole-array neighbourhood operations on such a structure means operating at any moment on

samples of the whole image. Neighbouring pixels are mostly in the same PE, and hence PE neighbour operations are greatly reduced; in the example, only one time in 16 is a PE neighbour operation needed for a nearest neighbour pixel access, and for a second nearest pixel the chance is 1 in 8, etc. There are no sheet boundary operations.

With this mapping, the full power of the processor array is not used if processing is needed only on a limited region of the image.

The greatest contribution to control overheads is liable to come from address calculation for indexing into the main data structure.

The memory requirements for crinkled mapping may be large, as temporary variables may be needed for whole images. Also, the fact that for a crinkled mapping a DAP matrix contains a sample of the whole image, means that the use of the fast I/O hardware on the DAP may be less efficient if internal buffers are not large enough to store a complete image.

2.3 Comparisons of Sheet and Crinkled Mappings

Some of the properties of the mappings can summarised:

	Local Area			Sampled Pixels	
	Name	Size	Mapped Onto	Sampling Interval	Mapped Onto
Sheet	Sheet	32 x 32	DAP Matrix	1 in 32 Linear	One PE
Crinkled	Local Area	N/32 x M/32	One PE	N/32 x M/32	DAP Matrix

N, M are the image dimensions

A table summarising some of the advantages and disadvantages is:-

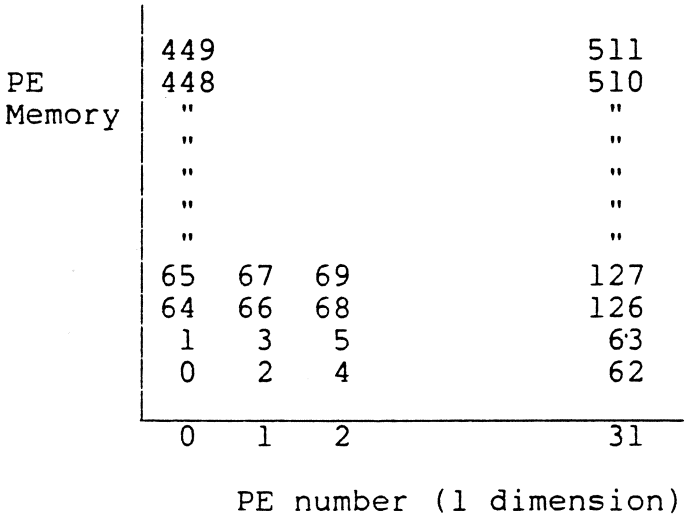
	Sheet	Crinkle
Necessary Processing		
Sheet Boundaries	bad	good (none)
Neighbour Accessing	bad	good
Limited Regions	good	bad
Algorithms	parallel	serial-parallel
Changing Resolution	bad	very good
Temporary Storage	small	large
Control Overheads	sheet boundaries	indexing arrays

"Algorithms" are discussed in section 8 and changing resolution in section 6.

2.4 Combined Mappings

As described above, crinkled and sheet mappings are merely the extreme cases of whole families of mappings. The two approaches can be mixed in either dimension separately, or between dimensions. Thus one dimension could be fully sheet and the other fully crinkled; or either dimension could be a mixture. An example of the latter would be to divide a dimension into "sheets" of sheet length 64 and then "crinkle" the length 64 into a processor array dimension of 32.

In one dimension this example is illustrated:



The numbers in this table are pixel numbers (1 dimension, with 512 pixels assumed).

The advantages and disadvantages of combined mappings are a mixture of those for the two approaches summarised in the previous section. This mixture can be a favourable one from the performance viewpoint, but mixing in the same dimension will usually be more complicated for the programmer.

3. LINEAR PROCESSOR ARRAY MAPPINGS

2D processor arrays can be used as 1D linear arrays; on DAP, the cost of linear shifts is roughly twice the cost of 2D shifts. Also some image processing architecture have a linear processor array to process one dimension in parallel and the other dimension is processed serially.

If the number of pixels in the parallel dimension is larger than the number of PEs, then the pixels can be crinkled in to the hardware. If the number of pixels is smaller, than two or more rows of pixels can be mapped across the processor array; for example, a 512 x 512 image on a 1024 PE array:

		Pixel Column Numbers				
		0	1	2		511
	0	0	2	4		1022
	1	1	3	5		1023
pixel	2	0	2	4		1022
row	3	1	3	5		1023
numbers	4	0				
	511					

The numbers inside the box are PE numbers. Thus an even PE contains all the even row pixel numbers of one pixel column.

If there is much more communication in one direction than the other, a linear mapping can be good if it is arranged that communication is between pixels in the same PE, and conversely the parallelism is across the non-communicating direction.

An example where a linear mapping can be appropriate is in Radar images where there may be

of the order of 1000 "range gates" which require largely independent prcoessing.

If communication is more symmetric between dimensions, a linear mapping will have an unnecessarily large communication overhead and may also require extra arithmetic.

4. CHANGING MAPPINGS

Often a single mapping is suitable for all pixel processing, but sometimes a change is desirable part way through.

During transforms such as FFTs data needs rearranging so as to bring together different data points. Another example (Reddaway 1987) familiar to the author is from radar processing. Here the 2D data has one dimension as range, and the other as spectral information leading to velocity via the Doppler effect. The first stage processing is FFTs in the spectral dimension, for which a linear mapping is best, with each PE allocated all data for one range; 1000 independent 64 point FFTs are performed with each FFT confined to a single PE. The next stage involves more symmetric 2D CFAR processing, such as cell averaging. For this it would be ideal to change from a 64 x 1 local area to a "crinkled" 8 x 8; however, a 16 x 4 local area is used as this involves less rearrangement and achieves most of the reduction in arithmetic and PE shifting.

5. DESCRIBING AND IMPLEMENTING MAPPING CHANGES

A powerful tool for describing a wide class of mappings of data arrays onto memory arrays has been developed by Flanders (1982). A more introductory document is "Data Mappings and routing for highly parallel hardware" by P.M. Flanders and D. Parkinson, August 1986, reference CM100 from AMT, 64 Suttons Park Avenue, Reading, UK.

Not only is a concise description achieved of mappings and of changes from one mapping to another, but when such a change is desired the concise change description can be used to invoke subroutines to implement the changes in the users' program. The tool, now known as Parallel Data Transforms (PDTs) applies mainly to data or hardware dimension sizes that are a power of 2.

6. CHANGE OF RESOLUTION

To change resolution can be advantageous in vision algorithms as the coarse resolution gives gross features and high resolution gives detail. This is sometimes known as pyramid processing. There is similarity to multi-grid methods for solving partial differential equations.

In moving to a coarser resolution the simplest technique is to average a 2 x 2 neighbourhood of pixels to form 1 new pixel. If needed, moving to a finer resolution can use interpolation or replication.

With a crinkled mapping the changes are extremely easy for resolutions finer than 32 x 32, as all the related pixels are in the same PE and no PE shifting is needed. If coarser resolution is needed, two approaches are possible: either leave the pixels spread across the processor array but with gaps, or else concentrate pixels towards a corner. In any case, the coarse resolution processing, including the resolution changes, is likely to be a minor part of the total as only one DAP matrix is involved; most of the work is with the multiple matrices of the fine resolutions.

Sheet mapping is much less suitable, as gaps immediately appear in going to coarser resolution, as well as PE shifting being involved.

Some vision architectures have proposed building the multiple resolutions into hardware via a pyramid structure. The practical efficiency of crinkled mapping makes that approach seem unnecessary.

7. MAPPING CHANGES DURING INPUT AND OUTPUT

So far no reference has been made to input or output. In a processor array it is both important to have a good data mapping and a non-trivial task to obtain it; it may take longer to rearrange data input in a poor mapping than to do the processing.

DAP has hardware inside the array to assist with the fast input or output of data with minimal impact on the processing. Associated with that hardware, DAP-2 has a programmable fast I/O sub-system that can achieve on the fly a wide class of data reorganisations between data in the array

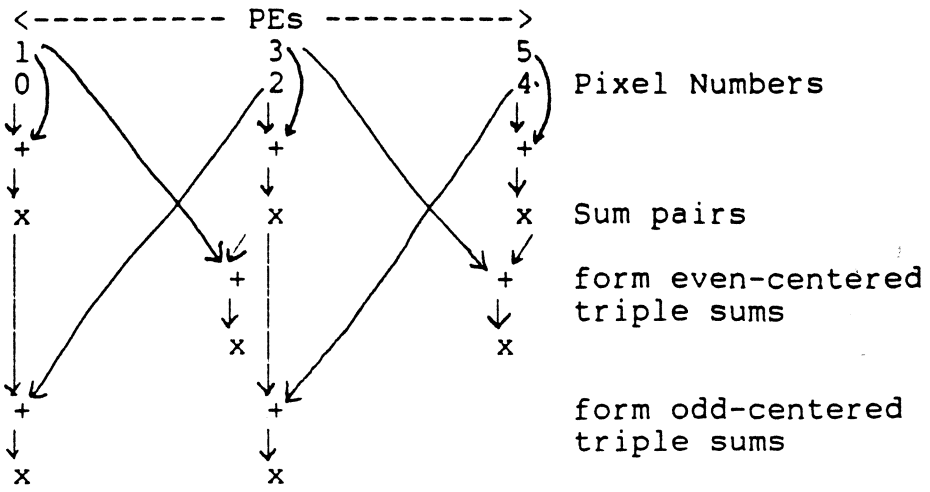
and data orderings required for input or output. Most reorganisations have similarities to PDTs (see section 5), but there is less restriction to powers of 2.

Most, but not all, reorganisations can be achieved 'free' in performance terms. Relating fully crinkled large images to an external raster is a problem because of the limited size of the data buffers in the sub-system, but usually organisations can be chosen that are optimal for DAP processing without being constrained by I/O requirements.

8. SERIAL-PARALLEL ALGORITHMS

It is often possible to get higher throughput from locally using a mix of serial and parallel techniques than from using the maximum local parallelism. In the former, parallelism is maintained by using higher level (rather than local) parallelism. Crinkled mappings lend themselves to such approaches, as the local area is being processed in a "relaxed" and fairly slow way, with power coming from processing 1024 areas (for full crinkling) spread over the whole array. This means that full advantage can be taken of serial optimisations for the local processing. The Sobel operator is an example treated in section 9. Another example is performing FFTs, where it is better to perform part (or all) of an FFT with many of the currently related points in the same PE rather than spread across different PEs.

Another simple example for illustration is cell averaging. If all sets of 3 consecutive points are to be totalled, a sheet mapping requires 2 PE shift and add operations (without allowing for sheet boundaries); a crinkled mapping allows pixel pairs to be summed (costing half an add per pixel) and then triple sums to be formed at a further cost of one add per pixel. For the mixed sheet/crinkled mapping of section 2.4, this looks like:



Thus the single step crinkling (64 into 32) has reduced the time for two DAP matrices from 4 adds and 4 shifts to 3 adds and 2 shifts. With allowance for sheet boundaries the improvement is bigger, and the reduction in shifting will be even more if a greater degree of crinkling is used.

This example illustrates how crinkled mappings can save arithmetic as well as neighbour shifts and sheet boundary movements.

9. SOBEL EDGE DETECTOR - AN EXAMPLE

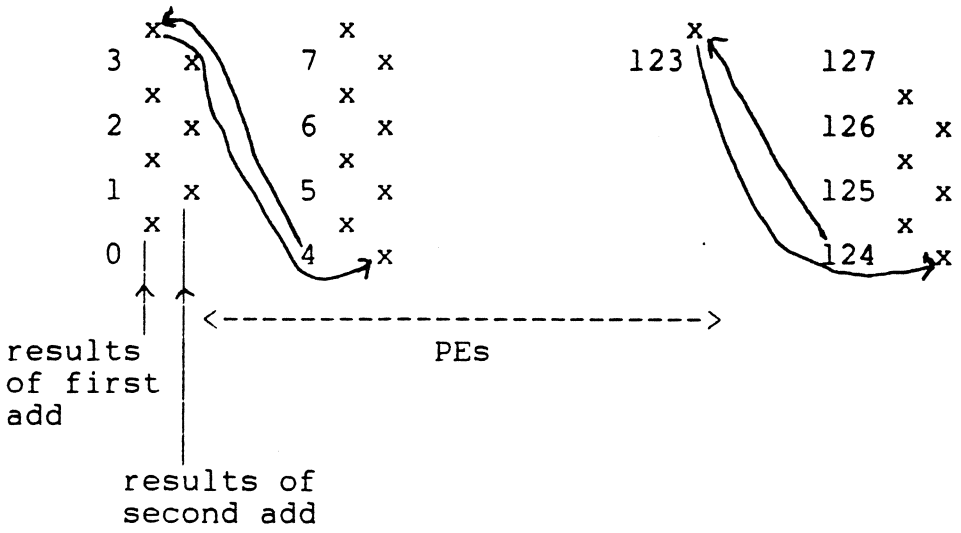
The Sobel operator is a special form of 3 x 3 convolution operator, with weights:

1	2	1
0	0	0
-1	-2	-1

It is often applied in both alignments in order to detect all edge angles. For simplicity of description only one alignment is assumed here. Careful use of intermediate results can reduce the number of adds per pixel to 3. The 1 2 1 pattern can be achieved by adding all nearest neighbours and then repeating the operation. The 1 0 -1 pattern can be achieved in a single subtract per pixel by combining second nearest neighbours.

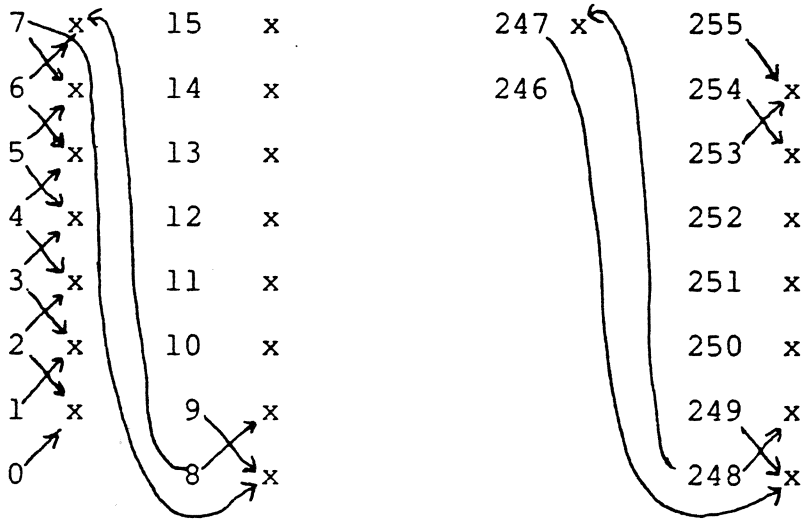
9.1 Crinkled Mapping

The 1 2 1 direction for an image size of 128 is illustrated:



The numbers are pixel numbers. It is mostly possible to overwrite old data. To deal with 4 matrices takes 8 adds and 2 shifts of 1 PE.

The 1 0 -1 direction for an image size of 256 is illustrated:



The numbers are pixel numbers of the first stage output. To deal with 8 matrices takes 8 adds and 2 shifts of 1 PE.

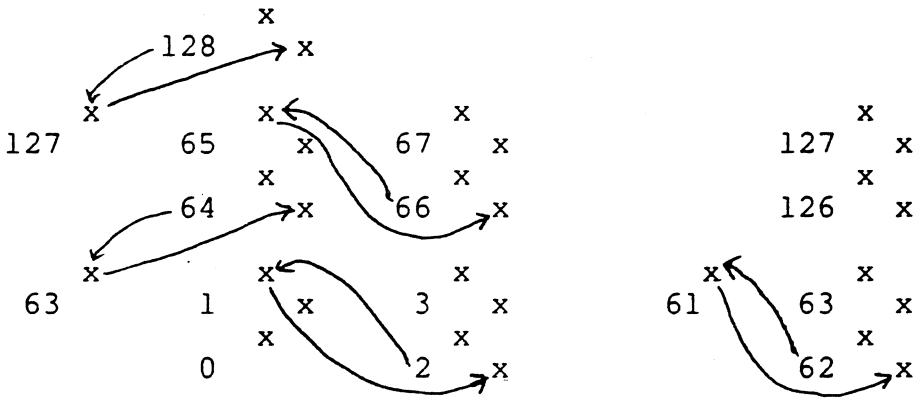
A 512 x 512 image fully crinkled to a depth of 16 x 16 requires a total of 256 x 3 = 768 matrix adds and 16 x (2+2) = 64 matrix shifts. On DAP these shifts can be combined with adds and hence cost an extra 1 cycle per bit-plane of data. (Basic adds cost 3 cycles/bit-plane). Thus the basic cost on DAP for 8-bit precision is:

(768 x 3 + 64 x 1) x 8 = 19000 cycles

Note that the entire neighbour overhead is less than 3%. To this should be added a control overhead and an allowance if precision is allowed to grow during computation. Reasonable allowances are 15% for precision growth and 10% for control, assuming careful assembly coding, or 25% extra in total.

9.2 Combined Mapping

The mapping assumed is of 64 x 64 sheets crinkled into 32 x 32 hardware. The 1 2 1 direction is illustrated:



This diagram has similarities to the corresponding one in section 9.1, with the 'basic' work for 2 matrices being 4 adds and 2 shifts. Extra operations are needed to deal with sheet boundaries: either 2 merge operations prior to shifting, or 2 extra shifts and activity controlled adds. The latter is assumed, and takes 4 cycles/bit-plane on DAP.

The 1 0 -1 direction is illustrated:

The basic work for 2 matrices is 2 adds and 2 shifts, with an extra 2 shifts and activity controlled adds for the boundaries.

A 512 x 512 image would require 128 pairs of matrices, each pair requiring 6 adds, 4 shifts and 4 boundary (shift and adds). At 8-bit precision this amounts to a 39000 cycles basic cost on DAP.

9.3 Sheet Mapping

The 1 2 1 stage requires 2 adds, 2 shifts and 2 boundary (shift and adds). The 1 0 -1 stage requires 1 add, 2 shifts and 2 boundary (shift and addds); there are 2 shifts associated with each add, and on DAP this means one of them takes 3 cycles/bit plane instead of 1.

A 512 x 512 image requires 256 of the above, which is a basic cost on DAP of 64000 cycles at 8-bit precision.

9.4 Sobel Summary

Collecting the above results, adding the 25% indicated in section 9.1 and taking a 100 nsec cycle time (DAP-3), gives for one application of the Sobel operator to a 512 x 512 8-bit image the following:

Mapping	Time for Sobel on 512 x 512 image
fully crinkled	2.4 msec
combined	4.9 msec
fully sheet	8.0 msec

The differences due to mapping are quite large.

10 PERFORMANCE ON SOME IMAGE ALGORITHMS

The table lists some algorithms, giving the best mapping (C= crinkle and S= sheet), time for 512 x 512 images on DAP-3 and estimates of the performance gain compared with the "opposite" mapping.

Table
Processing 512 x 512 images of 8 bits

Operation	Best Mapping	Time for DAP3 (msec)	Estimated Performance Gain
3 X 3 Sobel	C	2.4	3.3
3 X 3 Median filtering	C	16	up to 5
Histogram	_	40	-
Grey-scale normalisation	_	5	-
Summing or differencing	Same	0.6	-
Shrink or expand (per stage, 1 bit data)	C	~~0.6~~ 1.6	2-3
2 D FFT (8 bit growing to 18 bits)	S	70	1.2

Whole image neighbourhood operations (Sobel, Median, Shrink) are best crinkled. The Median time is based on a sorting approach described in document CM98 "Median Filtering on DAP" by S. F. Reddaway available from ICL or AMT. If shrink or expand operations are confined to limited regions a sheet mapping may be best.

Data mapping is irrelevant for whole image histograms, but local histograms will be several times faster with crinkling. Histogram techniques are described in Reddaway (1983) or Reddaway (1987). For summing or differencing, both images must have the same mapping. FFTs are best sheet mapped, as the early butterflies apply to the big data separations.

REFERENCES

Flanders, P. M. (1982). A unified approach to a class of data movements on an aray processor. IEEE Transactions on Computing C31-9, 809-819.

Reddaway S. F. (1983) DAP and its application to image processing tasks. In Image Processing Symposium, paper 6.3 RSRE, Malvern.

Reddaway S. F. (1987). Signal Processing on a Processor array. In Signal Processing, Les Houches 45 (eds. J. L. Lacoume, T. S. Durrani and R. Stora). North Holland Physics, Amsterdam.

Distributed Array Processors

S.F. Reddaway

Introduction

DAP is a highly parallel processor array with 1024 Processing Elements (PEs) which achieves high performance on a wide range of both numerical and non-numerical applications. It is available as an attached processor (initially with SUN or VAX hosts), or it can be embedded in OEM applications.

Background

The 1024 PE pilot DAP was completed in 1976 by my group in ICL; the first generation 4096 PE DAP product, on 330 PCBs, was delivered as a backend to an ICL 2900 mainframe in 1980. One site is Queen Mary College, London, where there is a DAP Support Unit that serves the national academic community. There is also a strong DAP centre in Scotland, at Edinburgh University Physics Dept. (in cooperation with ERCC) where there are 2 heavily used machines.

ICL developed a 1024 PE DAP-2 on 16 PCBs with the first of 12 prototypes delivered to RSRE in 1985. These were developed for both civil and defence applications, and in 1986 a new company (Active Memory Technology, or AMT Ltd) was spun-off from ICL to take over most of its DAP business, with ICL retaining defence interests. The 1024 PE AMT DAP500 series is expected later this year.

DAP Architecture and Programming

The DAP500 PEs have 32K or 64K bits (architecturally expandable to 1 Mbit) of RAM each, or 4 or 8 MBytes in all. The array has an (architecturally expandable) 32x32 2D structure in terms of both neighbour connectivity and highways across the array.

DAP is an SIMD machine which means it has a Single Instruction stream that can operate on Multiple Data spread over the PEs. Flexibility is greatly enhanced by each PE being able to exercise a local veto on array operations ("activity control").

The current programming language is Fortran Plus (previously known as DAP Fortran) which extends Fortran with operations on arrays. The atom of conventional processing and languages is the scalar variable. The atoms of DAP processing are scalars, vectors of 32 or 1024 elements, and 32x32 matrices. Indexed sets of these atoms can be used. There are powerful communication functions between scalars, vectors and matrices; these are natural in Fortran Plus and fast in execution.

Applications should be considered top-down; this usually results in both very good programmer productivity and fast execution. Trying to run or convert old code is less successful, although non-critical parts can be retained in the host or DAP.

As an illustration, consider the Fortran code:

```
      DO 10 M= 1, 32
      DO 10 N= 1, 32
      IF (A(M,N).GE.0) GOTO 10
      A(M,N)= A(M,N)+5
 10   CONTINUE
```

In Fortran Plus this becomes:

```
      A(A.LT.0) = A + 5
```

The Boolean matrix "A.LT.0" is used as an activity mask to control assignment to the matrix A. 1024 conditional jumps have become a single setting of PE activity.

Input/Output

As well as being able to do I/O via the host, DAP500 has a fast interface directly in and out of the DAP. This can achieve high speed (70 MBytes per sec), a wide range of data re-orderings "free" to the processing and opens up high data rate applications such as signal and image processing or graphics. A video board connection enables the DAP memory contents to be continuously displayed.

Applications

A significant amount of work needs to be expressed as array operations. This need not be arithmetic, but could for example also be:

- data movement
- table look-up
- searching
- sorting
- conditional operations

Potential applications cover a very wide area. A semi-random list is:

- matrix and field problems
- text processing
- sorting
- signal and image processing
- graphics
- CAD
- searching
- pattern recognition
- AI/Prolog

There is a growing library of algorithms and applications available.

A radar processing example will be discussed.

Performance

Performance is application dependent. Logical operations can be performed at over 1000 MOPS (Million Operations Per Sec), low precision arithmetic at hundreds of MOPS and floating point at around 10 MFLOPS.

Performance on total problems rather than inner loops is what really counts. DAP usually has an unexpectedly good overall performance.

Current Position and Future Developments

DAP-1s and DAP-2s are currently in the field, and the AMT DAP500 will be available this year. A simulator is available on SUN and soon will be on VAX. This allows programs to be developed without the hardware, although execution is a few thousand times slower.

For the future, DAP can be extended in terms of array or memory size, cycle speed and PE power.

INTERNATIONAL COMPUTERS LIMITED

C.E.O.(N.W.) ADVANCED SYSTEMS GROUP REPORT

REPORT NO. : ASG3

DATE : 5TH APRIL, 1971.

AUTHOR : K.W. WOOD

SUBJECT : A GENERAL PURPOSE ARRAY PROCESSOR WITH HIGHLY PARALLEL SYSTEM ARCHITECTURE

INTRODUCTION : A proposal is made for an Array Processor which will significantly speed up processing, and reduce software complexity in many fields of application.

Having established the functional concept, a choice can be made between implementation in either 'Conventional' or highly parallel architecture. The latter is chosen as the basis for this report.

This Report comprises Section A. Section B will cover more detailed exploration of the concept, and will be issued in due course.

K.W. WOOD

KWW/JR.

SECTION A - OUTLINE DESCRIPTION

A.1.0 CONCEPT

- 1.1 Hardware and Software Costs.
- 1.2 Electronic Peripherals.
- 1.3 Extracting a Common Theme.

A.2.0 FUNCTIONAL SPECIFICATION

- 2.1 Arrayed Data.
- 2.2 Scientific Processing.
- 2.3 Commercial Processing.
- 2.4 Selective Processing.
- 2.5 Sorting and Merging.
- 2.6 Class, Set or List Manipulation.
- 2.7 P.E.R.T. Network Processing.
- 2.8 Random Number Generation.
- 2.9 Support Functions.

A.3.0 SOFTWARE

A.4.0 PARALLEL SYSTEM ARCHITECTURE

- 4.1 Specification for Parallel System Architecture.
- 4.2 Outline Design.

A.5.0 TYPICAL PARAMETERS AND PERFORMANCE

A.6.0 INTEGRATED FILE STORE

A.7.0 TECHNOLOGY

A.1.0 CONCEPT

A.1.1 Hardware and Software Costs

The prevailing situation is one in which digital computer hardware is becoming available with rapidly improving cost-performance.

These improvements are currently available from M.S.I. packages: an even more dramatic improvement is promised with the advent of LSI.

Unfortunately, no such revolution is promised in the field of Software production.

This imbalance has stimulated the study of the Electronic Peripheral concept at CEO(NW) and UMIST.

A.1.2 Electronic Peripherals

An Electronic Peripheral can be defined as a dedicated processor which is attached to a general purpose system in a peripheral-like manner. The dedicated processor provides improved processing rate and/or minimised software requirement in a specific application or class of application.

Several such devices have been proposed at both CEO(NW) and UMIST.

A further relevant proposal - The Content Addressed File Store - has been made by RADC.

A.1.3 Extracting a Common Theme

Having established a repetoire of Electronic Peripherals, the question is now posed as to whether one could extract any principal, recurrent, hardware features, and combine these to form one machine with a wide range of application.

The following table (A.1.2) arrays Hardware Features against Application Areas, and indicates where the Feature is relevant (by a tick).

The proportion of ticks is about 75%. It will be noticed that one basic Feature is Array Processing.

Fig. A.1.2 TABLE OF HARDWARE FEATURES -v- APPLICATION AREAS

Area / Feature	Scientific	Commerical	OR & PERT	File Enquiries	Operating Systems
Arrayed Data	✓	✓	✓	✓	✓
Scientific Arithmetic	✓		✓		
Commercial Arith. & Edits	✓	✓	✓	✓	✓
Selective Functions	✓	✓	✓	✓	✓
Sorting & Merging	✓	✓	✓		✓
High Rate File Store	✓	✓	✓	✓	✓
Basic U.D.R.N.G.	✓		✓		

It is thus proposed that a machine with the stated features will give a high processing performance, and minimisation of Software over a wide range of applications.

This machine will be referred to as a General Purpose Array Processor.

A.2.0 FUNCTIONAL SPECIFICATION

A.2.1 Arrayed Data

An Array of Data can represent a Vector, List, Matrix, Table, Row, Column etc.

The operands comprising an Array are loaded into store in a regular, pre-determined manner.

A.2.1.1 Scientific Data

Figure A.2.1/1 represents a typical method of loading a 'Matrix' into store, and shows how any row or column of the Matrix may be specified as an operand array.

A.2.1.2 Commercial Data

Figure A.2.1/2 shows how the concept can be extended to handle commercial data. Thus several records, loaded contiguously into store, form a Table.

A Row represents a Record.

A Column represents equivalent data in all Records.

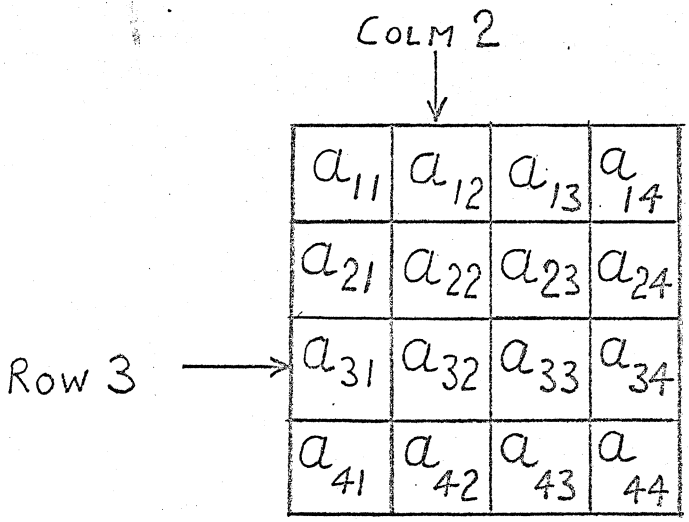
A.2.1.3 Use of Striding

It will be noted from the above that the facility for 'Striding' has been incorporated as a basic feature. This concurs with 1900 Array Processor, and New Range Descriptor, practice.

C.D.C. Star Vector Instructions, in general, only operate on contiguous operands. This is to enable a high operand stream rate to be maintained from the Store System. Low efficiency instructions are provided to transfer

FIG A 2.1.1, DATA ARRAYED AS A MATRIX

MATRIX



LAYOUT IN CORE

(COL^M BY COL^M)

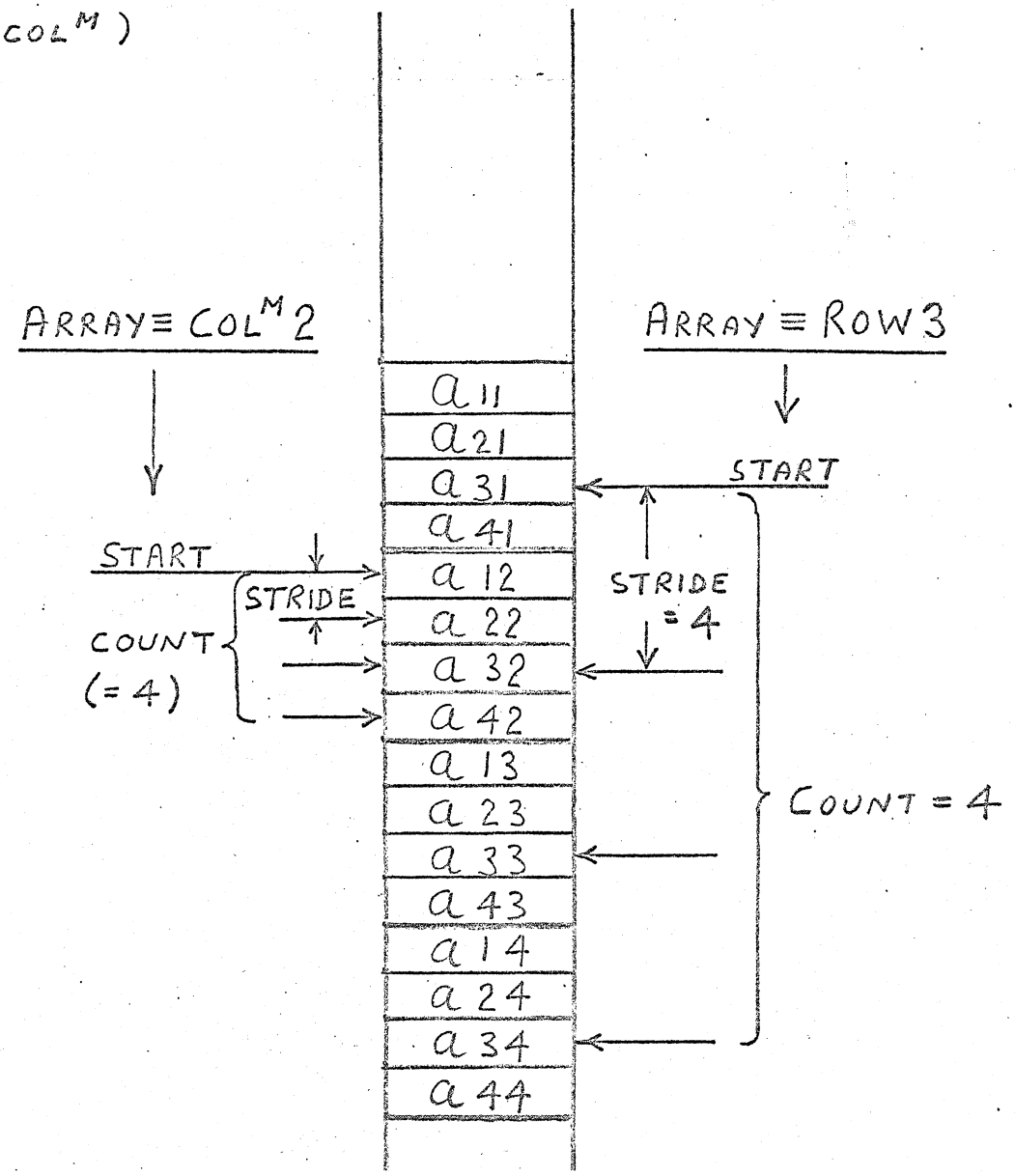


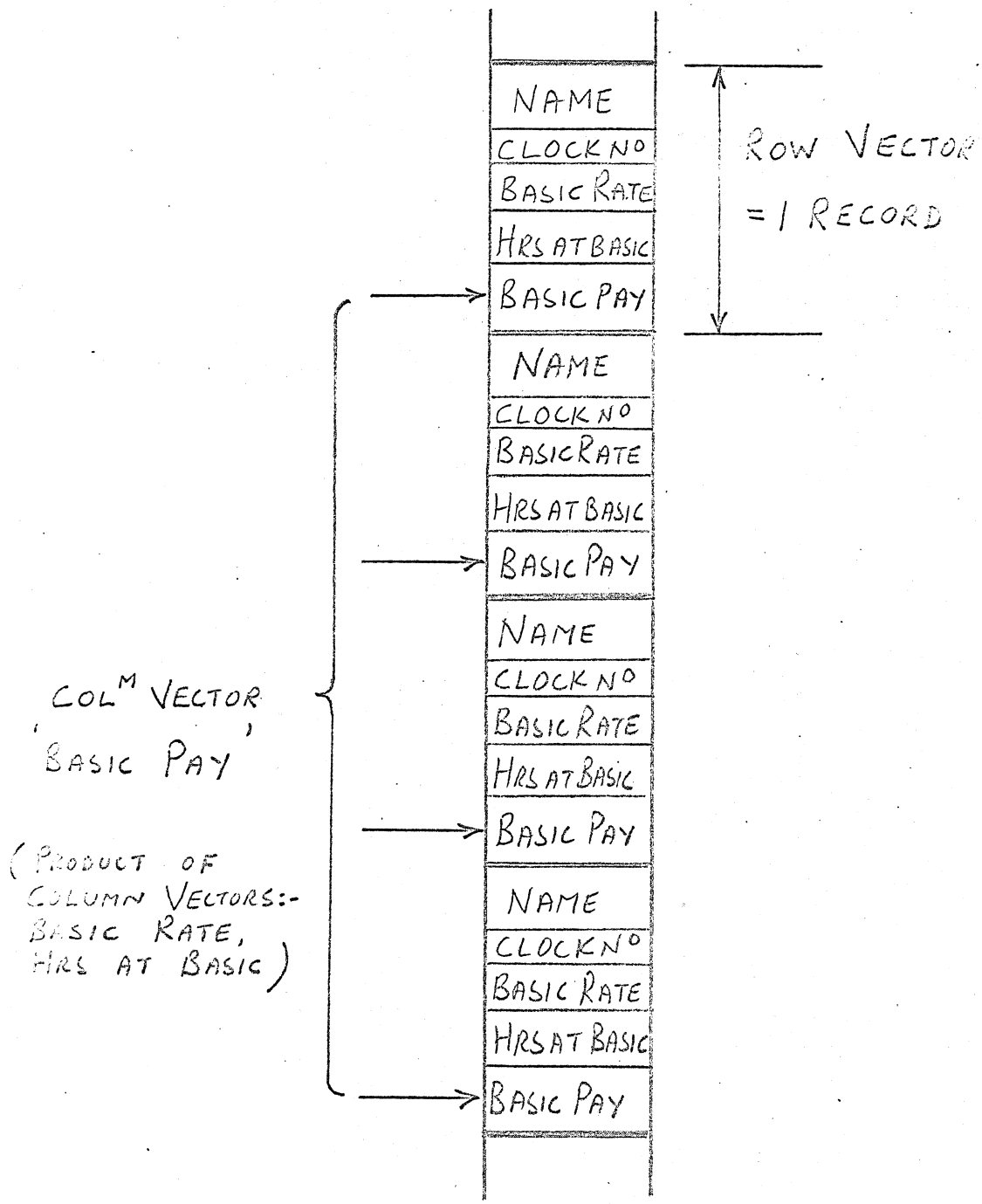
FIG A2.1.2

DATA ARRAYED AS A COMMERCIAL TABLE

TABLE:-

		A	B	A × B
NAME	CLOCK NO	BASIC RATE	HRS AT BASIC	BASIC PAY

LAYOUT IN CORE:-



rows to columns before processing by high efficiency Vector Instructions.

However, the incorporation of Striding facilities in Vector Instructions are considered to be highly desirable.

The benefits are:

Ease of programming.
Feasibility of Commercial Data Array handling.

As will be seen later, the proposed Parallel system architecture has been chosen to give a high operand stream rate on both Contiguous and 'Striding' arrays.

A.2.1.4 Sparse Arrays

Where an Array contains a preponderance of zero elements, then the data is stored and processed in 'Sparse' form. Only the non-zero elements are stored together with an indication of their position in the Array.

The position can be indicated by one of at least 3 methods:

(a) A Boolean Vector with 1 bit per element of the full array.
0 means zero value
1 means non-zero value
This method is used in 'Star'.

(b) A Vector of position numbers - 1 position number per non-zero element.

(c) An interleaved Vector of position number and value.
This method is used in the 1900 Array Processor.

Choice of method could depend upon source data format, programming language, user preference, hardware structure.

A.2.1.5 Control Vectors

These are boolean vectors, associated

with Noraml (non-sparse) Arrays, and having 1 bit per element of the array.

0 means do not process the element

1 means process the element

The bits of the Control Vector are set by some previous attribute test, and enable the elements of an array to be selectively processed.

The use of Control Vectors will be fairly essential to Commercial Array Processing.

An alternative technique in the use of zero operands, i.e. the relevant numerical factor, is forced to zero if processing is not required.

This second method is less efficient in 'mill' utilisation.

Use of either technique can simplify programming by the elimination of some conditional branches.

A.2.2 Scientific Processing

Operands : Floating Point 32, 64 or 128, N.R. Compatible.

Arrays : Normal or Sparse Vectors or Matrices

Function Groups:

Fixed to Float conversion and vice versa.

Data Movements and Transpositions.

Simple Arithmetic Operations on Vectors and Matrices - including Inner and Outer Products.

More Complex Operations using Polynomial type Evaluation.

'Statistical' Functions - Moving Averages, Correlations, Convolutions.

The 1900 Array Processor Instruction Set (see Section B.2.2) is a good starting point for more detailed consideration.

The ideal Instruction set should give ease of compilation, and efficient execution of programs written in an agreed Array Processing type Language - possibly A.P.L.?

A.2.3 Commercial Processing

Operands : Byte String, Packed Decimal Digit or Binary.

Arrays : Tables formed from Contiguous Records.

Data Format: Specified by Usage and Picture as in COBOL.

Variable length fields specified by Keys as in 1900 COBOL.

Functions : Move, (including Editing and Radix Conversion).

Add

Subtract

Multiply

Divide

The above facilities should make for rapid manipulation of data in Commercial Applications, and during the input/output phases of Scientific applications.

Again, final choice of instruction set will be influenced by the target User Language(s).

A.2.4 Selective Processing

Operands and Arrays:
Scientific or Commercial.

Functions :

Search for: Maxima
Minima
Equality
Greater Than
Less Than

Results would be:

Numerical Scalar

Numerical Vector

Boolean Vector (c.f. Control Vector).

A.2.5 Sorting and Merging

Operands and Arrays: Scientific or Commercial

Functions:

(a) Produce Grade Vector from Key Vector (Ascending or Descending Order. The Grade Vector gives the positions of the Keys when sorted. Could be cumulative to resolve 'ties'.)

(b) Transpose Data Vector by Grade Vector. (Data is moved into the Sorted Order dictated by the Grade Vector).

(c) Produce Merge Vector from two Sorted Key Vectors. (The Merge Vectors give the positions of the Keys when merged to form a combined sorted block).

(d) Transpose Data by Merge Vector.

Sorting operations of any required complexity can be built-up using the above basic functions.

A.2.6 Class, Set or List Manipulation

Operands and Arrays : Commercial

Functions : Zero
Load
Lose
Head Gain
Tail Gain
Converse
Convert

The above are the basic requirements for an O.R. Simulation language such as C.S.L.

Other functions could be added to facilitate other forms of List Processing, e.g. for Artificial Intelligence work.

A.2.7 P.E.R.T. Network Processing

Operands and Arrays : Commercial

Functions :

Performs the logical progression phase of PERT processing.

Given a PERT Network defined by Preceding and Succeeding Event Columns (Vectors), and a Boolean Vector of Activities completed, the function examines the Network logic and determines which new activities can be initiated.

This function could also be used to control the processing of structured data, e.g. building up a product cost from component prices, quantities, and 'Assembly Logic'.

This function could also be extended to facilitate logic simulation.

A.2.8 Random Number Generation

It is proposed to include a basic Uniform Distribution Random Number Generator as a 'Special Register'.

This facility, used in conjunction with the Scientific Array Processing functions, would enable random number streams to be generated with any required distribution.

A.2.9 Support Functions

An appropriate selection of functions would be required for general control and organisation.

A.3.0 SOFTWARE

There are 3 basic methods by which Array Processing functions can be brought into use by a User Program:

(a) By the use of 'Intelligent Compilers which examine normal high level language statements and decide whether loops exist which can be transformed to Array Processing Functions.
This method would enable existing programs to use A.P. facilities whenever appropriate, but represents a major undertaking for Compiler writers.

(b) By the use of routines, written in Array Processor code, and called up from High Level Language Programs.
Library routines could be replaced by Array Processor versions, and thus the User would automatically get the benefit of Array Processing whenever he called up such a routine.
However, in other cases, the User would have to amend his program to call up special A.P. routines.
This method would entail the writing of many routines in Array Processing Code, and the amendment of many existing programs.
However, it can be tackled in a fairly continuous manner, and does not possess the inherent difficulties of method (a).

(c) Use of a High Level Language which is closely aligned to the Array Processing functions of the Hardware.

The I.B.M. language A.P.L. (A Programming Language) is an attractive possibility.

This would necessitate a special Compiler of course, but not a particularly intelligent one!

The advantages of A.P.L. are:

Direct handling of Arrays

Comprehensive repetoire of Operators

Minimisation of conditional branches (by the use of Selective Array Functions).

These features enable programs to be simpler and more compact.

Alternative (c) is considered the most attractive long-term proposal.

Subsequent refinements of the proposed Function repetoire will tend to be based on this view.

A.4.0 PARALLEL SYSTEM ARCHITECTURE

So far, the General Purpose Array Processor has been defined by outline functional specification.

This functional specification could be fulfilled by a machine working in Operand by Operand mode, c.f. the 1900 Array Processor.

However, the reiterative nature of Array Processing stimulates the concept of parallel processing of elements within an Array.

The end result of such a system will be an even greater improvement in performance. Since the machine is general purpose, it should enable the whole spectrum of User programs to be significantly up-lifted in performance.

The attraction of a Parallel System from the engineering point of view is that the reiterative nature of the logic could provide stimulus for L.S.I. development.

This document is therefore aligned to implementation in a Parallel Architecture.

A.4.1 Specification for Parallel System Architecture

The following target requirements have been set:

(a) High Data Throughput Rate from Storage System.

Must apply to both 'Contiguous' and Striding Arrays.

(b) High Processing Rate.

(c) Modular

To enable cost/performance to be balanced to specific customer requirements.

(d) Re-configurable

In the event of failure in one storage or processing module, the system should be kept running.

(e) Parallel Structure Invisible to Software

No additional software should be required for mapping data into the storage system, nor for manoeuvring data during processing.

(f) Normal Interface to any Parent G.P. Computer System

The G.P.A.P. and its integrated storage system should be usable as standard 'blocks' in a N.R. Multi-Processor System. Thus, if no A.P. programs are imminent the A.P. Store could always be temporarily redeployed as additional System storage.

A.4.2 Outline Design

See Figure A.4.2.

(a) Storage System

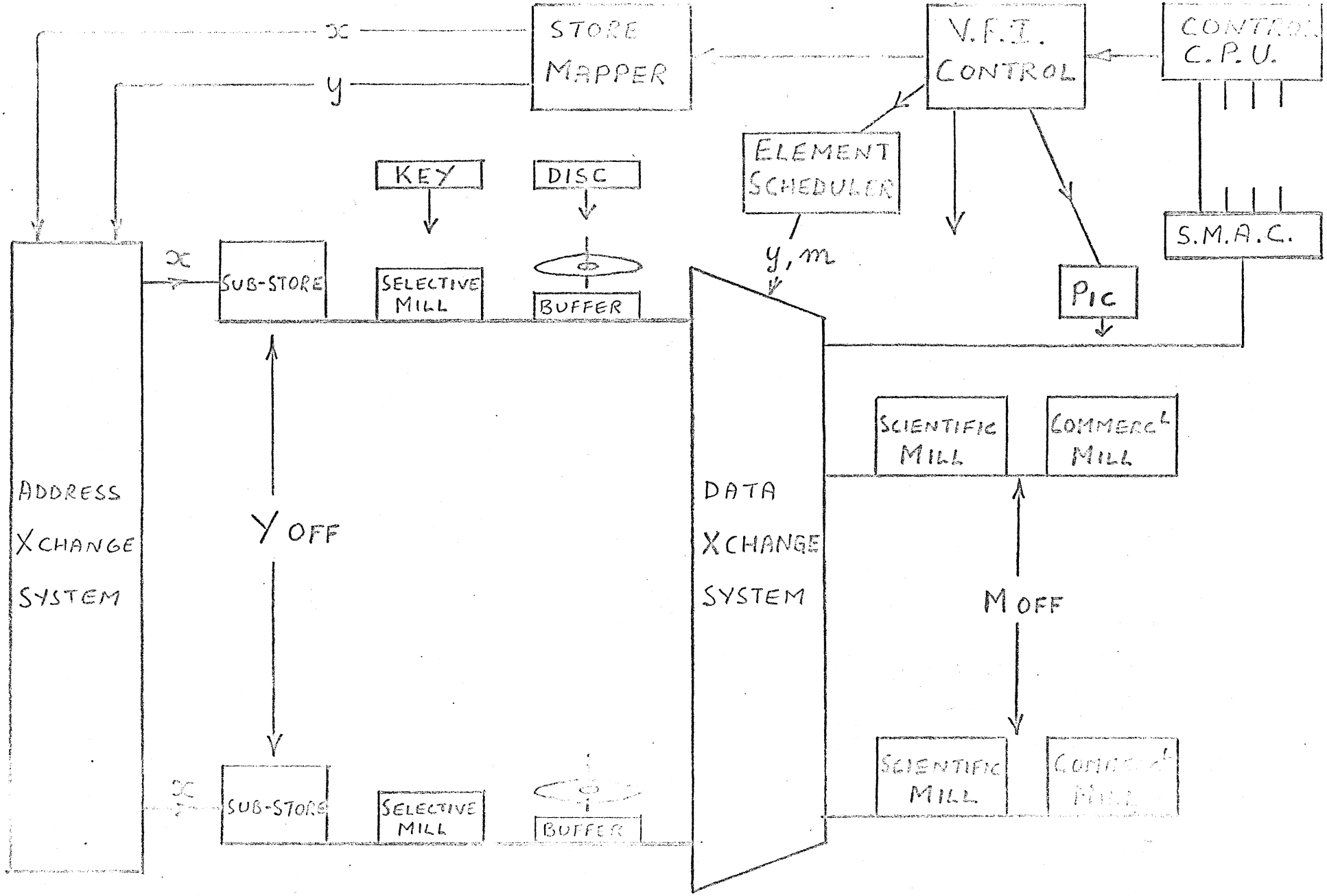
The total store consists of many (Y) sub-stores - each individually addressable. This concept is economically viable with semi-conductor store units.

(b) The Store Mapper

The store system is effectively two dimensional and arrays are mapped into the system in such a manner as to facilitate fully parallel access except in a few well defined cases.

The mapping algorithm is based on the choice of a prime number for Y.

FIG A4 - BLOCK SCHEMATIC



This eliminates bunching of operands on any sub-store highway, except when the Striding Factor is a multiple of Y. These represent the few well defined cases.
The mapping is performed entirely by hardware - operating on the parameters, Starting Address, Striding Factor and Count, as specified by the typical Vector Instructions.
The Store Mapper generates a pair of co-ordinates for each element in the Array
x = Address within Sub-Store
y = Sub-Store Number.

(c) The Address Exchange System

Routes each x value to the appropriate Sub-Store - as determined by the corresponding y value.

(d) The Selective Mills

Enable Selective Functions to be carried out directly on the Sub-Store Highway.

(e) Disc Access Channels

See Section AG.

(f) Data Exchange System

In a typical Vector Function, the two operands and result will occur on different Sub-Store highways.
A Data Exchange is therefore necessary to route data to and from the Allocated Mill.

(g) The Element Scheduler

This unit keeps a tally of element numbers, corresponding Sub-Store numbers and Mill numbers, and keeps events in their proper sequence.

(h) Scientific and Commercial Mills

These carry out the appropriate processing functions.
There is possible scope here for implementing a combined Mill using Cellular Logic Techniques.

(i) Picture Control

Controls the processing of Commercial data according to the 'Picture' information specified by the Instruction.

(j) V.P.I. Control

Controls the processing of Vector Instructions.

(k) Control C.P.U.

This would provide the Support Functions for general control and organisations. In the case of a parent N.R. System, it could be a minimal N.R. C.P.U. and thus able to communicate with the rest of the system.

(l) SMAC

Again in the case of a parent N.R. System, could enable the A.P. store system to be accessed by the rest of the system.

A.5.0 TYPICAL PARAMETERS AND PERFORMANCE

The following are approximate only (±?) and are based on extrapolations from the 1900 Array Processor predictions.

Target Performance	125 x Atlas 50 x 1906A
Time to Process 1 Element	32nS
Number of Sc. + Comm. Mills (at 256nS per element)	8
Number of Sub-Stores (at 250nS Cycle)	61
Relation to other machines	5 x STAR (with Vector Instructions) 30 x 6600 7½ x 7600 32 x 360/165

A.6.0 INTEGRATED FILE STORE

The proposal for the integral connection of a Disc File Store has been inspired by the RADO project C.A.F.S. (Content Addressed File Store).

Greater exploitation of the high processing rate of the G.P.A.P. system can be made if a significantly improved data rate from File or Backing Store is available.

The proposed method, as can be seen from Fig. A.4.2, is to provide each Sub-Store Highway with a Read/Write channel to the Disc - all channels being capable of Reading (or Writing) simultaneously.

Thus, 61 words are accessed in the time normally taken to access 1 word, and this represents the gain in bulk data transfer rates.

Where data is required to be retrieved by Content Addressing, the comparison against the specified Key can be made using the Selective Mill on each Sub-Store Highway. Thus 61 Keys can be compared simultaneously.

This principle is used by C.A.F.S.

A.7.0 TECHNOLOGY

A.7.1 Chips

Trial designs could initially assume:

(a) Logic Chips with complexity equivalent to present M.S.I. TTL functions, but implemented in Schottky, or MECL 3, as dictated by performance.

(b) Storage Chips with present (or imminent) complexity.

(c) Possible use of MOS logic chips.

Areas of logic amenable to larger scale integration should then become apparent.

A.7.2 Packaging

Basic packaging could be based on ICL Microwiring Techniques.

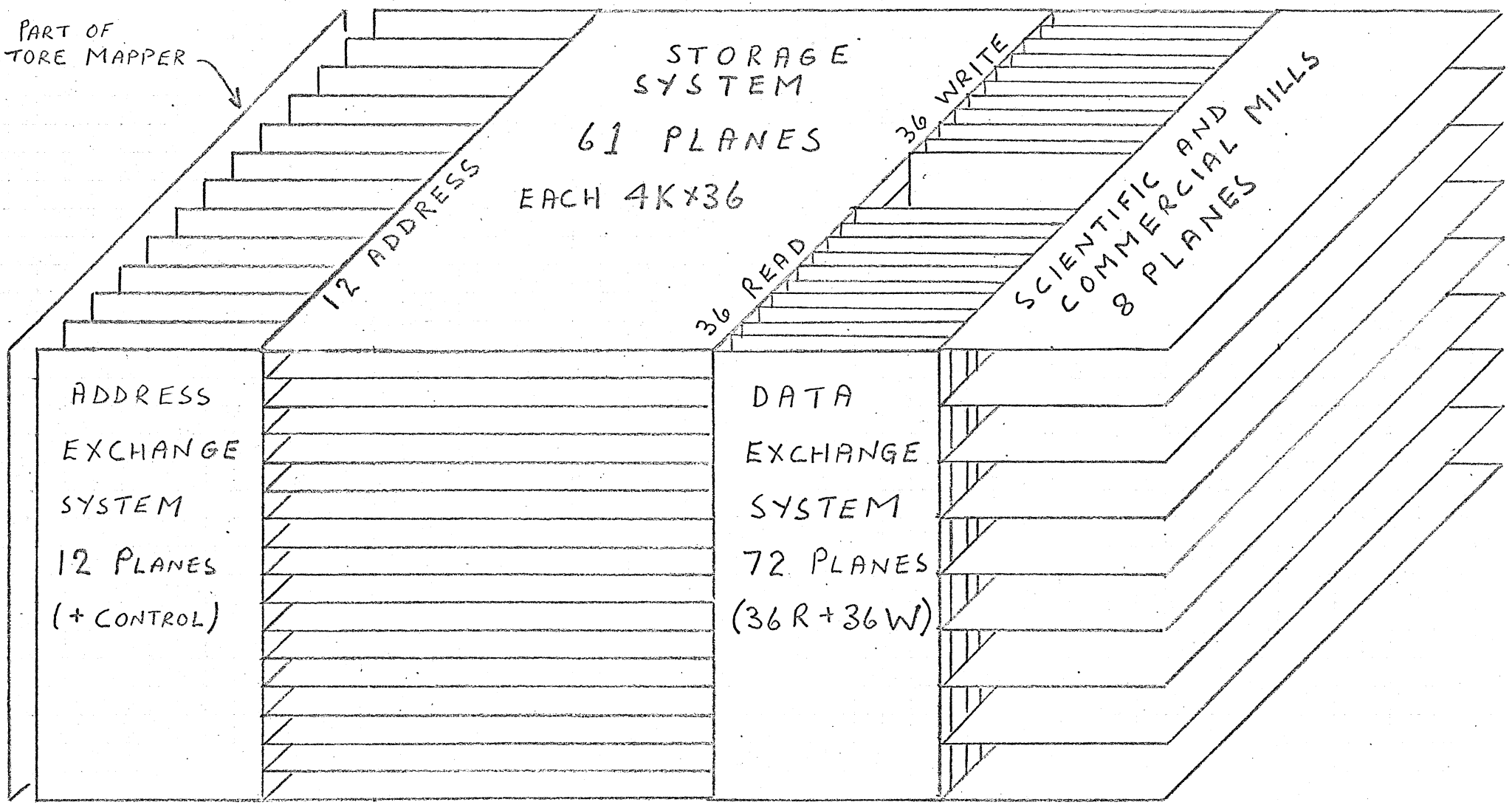
However, initial studies show that the extensive data highways could be difficult to implement in Planar or Bookcase packaging.

A more favourable topology is as shown in figure A.7.2.

The planes represent maximum density of logical interconnection. Minimum interconnection is required between planes of a given Stack.

The bulk of data highway connections occur at the points of intersection between vertical and horizontal planes.

A 7.2 TOPOLOGY



CIRCULATION

C.E.C.(N.W.):

MR. W.J. TALBOT
MR. C.D. MARSH
MR. G. HALEY (12)
MR. D.C.P. PORTMAN
MR. C.P. BURTON
MR. R.H. ALLMARK
MR. K.C. JOHNSON
MR. K.C. EVANS
MR. A.B. COOPER
MR. G. THOMAS
MR. K. HOWKER

C.P.O.:

MR. A.R. BAGSHAW
MR. D.T. CAMINER

BUSINESS PLANNING:

MR. A.C.D. HALEY

R.A.D.O.:

MR. G.G. SCARROTT
MR. R.W. MITCHELL

GROUP H.Q.:

MR. P.R. BRIGGS
MR. R.J. ROUGHLEY

THE DAP

AND

SUPER PERFORMANCE COMPUTING

THE ROUTE TO LOW COST SUPER PERFORMANCE COMPUTING AND THE ROLE OF THE DAP

A position paper by G Manning, CEO of Active Memory Technology

April 1990

1. INTRODUCTION

Computers are purchased to solve users' needs. Some tasks can be met by a modest single processor system like those found in the average home computer. Others require systems with thousands or even millions times more power. These tasks cannot be met by even the most powerful single processor system. Physical laws limit the speed of such uniprocessor systems. For such compute intensive tasks the only way ahead is to use many processors working efficiently on a single task - ie to use parallel processing. A system of 100 times the power of a home computer can be produced in a single machine but an alternative is to use 100 processors of a home computer power - providing that the software will enable the 100 units to work together with high efficiency. If the aim is a million times that of a home computer, then no single system can be made to satisfy this task and the only route forward is parallel processing.

Figure 1 shows the options available in a plot of the number of processors against the power of each processor. The system power is shown in "home computer units". The first 30 years of computing development can be represented as a movement along the lower edge of this plot. Single processor systems of increasing power are often labelled "workstations", "minicomputers" "super-minicomputers", "mainframes", and finally "supercomputers". The quest for increased supercomputer performance has driven these systems to become parallel and hence modern supercomputers have 4, 8 or even 16 processors because they are close to the physical limit of a uniprocessor. This physical limit is represented in figure 1 as a brick wall beyond which one cannot pass.

For most users there are two important questions. Can a computer system perform my task? What is the least cost system to do this? System cost must include both the hardware and software costs and often price-performance and ease of use dominate. For some users there are other requirements such as compute power per unit volume, per unit weight or per watt.

The task for the computer manufacturer is that of how to provide the user requirements at the lowest cost and in the most user friendly way.

2. WHAT ARE COMPUTE INTENSIVE PROBLEMS?

In many cases compute intensive problems are "data parallel" and relatively simple tasks are performed billions of times on different data. An example is processing TV pictures containing 250,000 picture points in each frame and 30 frames per second, making 7.5 million picture points per second or 27 billion per hour. There is no reason why a thousand or even ten thousand picture points cannot be processed in parallel. It is also worth noting that the data to be processed is of limited precision and can typically be represented by say 8 binary bits. There is no need for 32-bit floating point arithmetic for this task.

A second example of a data intensive task from an entirely different area is searching a database, of say 50 gigabytes containing 5 million articles, for those containing some selected words. Again this can easily be undertaken in parallel and the data is of limited word length - a letter can be represented by 5 to 8 bits depending on the size of the alphabet.

The two examples given above can be generalised. Compute intensive tasks are always parallel. If a task runs for tens or hundreds of hours on a modern computer system there must be some sections of the program that are executed millions or billions of times since no programmer, or programming team, can write enough code to keep the system busy that long if each line of code is executed only once.

Are there compute intensive tasks that cannot be tackled efficiently by say 1000 processors in parallel? The answer is "yes" but they tend to be rare. Even those we think today are not possible may be possible if new algorithms are developed - this is an important research area.

3. IS IT DIFFICULT TO PROGRAM A PARALLEL COMPUTER?

This is a simple question but the answer is complex. There are many forms of parallel computer. The most common are called SIMD and MIMD in the jargon of today. SIMD stands for single instruction multiple data - all the processors obey the same instructions but work on their own data. Hence the user has to write only one program. All the processors are locked in step - there are no problems of data synchronisation. SIMD systems are simple to use and it is easy to make a massively parallel system efficient, particularly on data parallel problems. Thus they provide outstanding price-performance. Hence when a problem is suitable for a SIMD system it should be used. A large fraction of compute intensive problems map well onto SIMD systems.

MIMD stands for multiple instruction and multiple data. Each processor runs its own separate program and operates on its own data. These systems tend to be built from more powerful processors but with a smaller number because it is difficult to keep the system efficient as the number is increased. They are often called coarse grained systems. MIMD systems are more difficult to program than SIMD and are more expensive for a given compute power. In spite of this, most parallel computer companies use this technology with the building blocks being commercially available floating point microprocessors. Most systems sold have fewer than 100 processors, although a few larger systems have been built.

Figure 2 shows where SIMD and MIMD systems fit into the future computer scene. Both systems are capable of producing extremely powerful systems and are competitors to todays supercomputers.

4. THE AMT SOLUTION - THE DAP

AMT is using VLSI technology to drive down the cost of compute power. This technology does one thing superbly well - it replicates simple units in large numbers at very low cost. It is this fact that has driven down the cost of memory and doubles the storage capacity of memory chips at about a one year interval. AMT is replicating a simple compute engine in very large numbers to produce a very powerful system. We have 64 simple processors on a chip of similar size and cost to a single 32 bit microprocessor. The DAP makes very efficient use of silicon and allows an increasing scale of integration.

AMT stands for Active Memory Technology. The name describes an important aspect in the AMT solution - it is to integrate memory and processors. For most powerful compute systems the designer has to struggle to provide adequate "bandwidth" to supply data from memory to the processors. The more powerful the processors, the higher the bandwidth has to be, and in many systems it is this that proves to be the bottleneck that limits the useful compute power. The AMT solution is to be massively parallel in the solution to this problem. Rather than a data bus of 32 bits width as used in nearly all computer systems, the DAP data bus is 1024 bits wide or 4096 bits wide. Each simple processor has its own memory and its own path to that memory. The close integration of memory and simple processors, "active memory", exploited in very large numbers enables VLSI to be used to provide low cost compute power and very high internal bandwidth to memory within a single elegant architecture.

The DAP has traditionally used the simplest possible building block - a one-bit processor. It is the ultimate RISC (reduced instruction set computer) system. These processors are connected into a square array with nearest neighbour connections plus row and column data paths. The square is 32 x 32 giving 1024 processors in the smallest system (DAP 510) and 64 x 64 (4096 processors) in the DAP 610. This processor-to-processor connectivity is another important design feature of the DAP. AMT provides in hardware efficient solutions to the most commonly required connectivities. Other, less frequently required connectivities, are provided by user friendly software. The overall DAP system is therefore an elegant and very cost-effective compromise based upon our 15 years of experience. For the AMT designers our initial offering was a third generation product - this experience is unique in the parallel computing industry.

User access to computer hardware is through software - it is therefore this medium that is judged by the typical user. Again the DAP systems software is a well proven product. The AMT strategy is to make the DAP as transparent as possible to the users. They access the DAP through a VAX or Sun host and do not have to know about the DAP operating system. AMT provides tools, compilers, languages and libraries to aid the users in programming those parts of their existing programs that are reprogrammed to run in the DAP. Only the compute intensive kernels of code that are executed millions of times need be transferred. DAP users find the system easy to program and the code produced is very efficient.

Figure 3 shows where the current DAP 510 and DAP 610 are positioned compared to other systems. They are not as powerful as supercomputers but are a great deal cheaper. They provide very much improved price-performance.

5. WHERE DOES THE NEW DAP/CP8 FIT?

AMT's new product the DAP/CP8 adds 8-bit co-processors to the DAP. One 8-bit co-processor is added for each one-bit processor. Hence the DAP/CP8 model 510C contains 1024 one-bit processors plus 1024 8-bit co-processors. The DAP/CP8 model 610C has 4096 of both the one-bit processor and the 8-bit co-processors.

The DAP system software decides how to use the dual 1-bit and 8-bit processors - this is completely transparent to the user. The 1-bit processors are used for memory access, for data movement, for input and output, and for Boolean or logic tasks. The 8-bit co-processors are used for complex arithmetic such as floating point operations. The improvement in performance for a given application depends upon the mixture of instructions used and can be 1000% better than the corresponding DAP system without the co-processors.

The floating point performance of the 510C model is up to 140 MFLOPS and the 610C up to 560 MFLOPS. This provides 32-bit floating point price-performance as low as $700 per MFLOP. The 8-bit integer performance is up to 5000 MIPS for the 510C and 20,000 MIPS for the 610C giving a price-performance of $20 per MIP. These price-performance values provide a dramatic improvement over those offered by conventional systems. The DAP/CP8 range is up to 700% better in price-performance than the DAP range without the co-processor, which was already competitive with the most cost-effective conventional systems.

Figure 3 shows the power of the new CP8 range. The DAP delivers super performance computing for some applications. For problems that require only limited accuracy of data (eg. 8-bit integer) it will substantially outperform any supercomputer. Examples of such tasks were given in section 2 above. The DAP performance increases as the accuracy required decreases - this is not true for supercomputers or for parallel systems built from 32 or 64-bit microprocessors.

6. SOFTWARE COMPATIBILITY BETWEEN DAP/CP8 AND DAP RANGES

Software written for a DAP will run unmodified on a DAP/CP8 - only recompilation is required. AMT has released a new compiler that allows the use of arrays of any size and automatically maps onto either a 32 x 32 or a 64 x 64 array. Hence user source code will run unmodified on a 510, 610, 510C or 610C. These systems have a range of 40 in peak compute power.

DAP systems have been in use for more than a decade. There is a massive platform of application software produced largely by academic institutions that demonstrate the effective use of the DAP in a wide range of applications areas. There are about 1,000 publications on this work. AMT has a customer base of 75 installed units with approaching 1,000 active users. The source code compatibility between the new DAP/CP8 range and the current DAP range ensures that the interests of the current users, and the very large asset of previous application code, are maintained.

AMT policy is that all hardware improvements must be source code compatible with existing machines. One of the great strengths of SIMD systems, and the DAP in particular, is that they are scalable - software developed on a small system will run unmodified on larger systems and the performance will scale as the number of processors is increased. The new CP8 range extends that scalability to include improvements in the power of the processors. DAP users can safely assume that future AMT systems will be compatible with their existing software and that the development path shown in figure 3 is readily available to them.

7. OTHER FACTORS - COMPUTE POWER IS NOT THE ONLY IMPORTANT ASPECT

The sections above have concentrated on the quest for increased compute power and the ease of programming. These are very important factors but they are far from the total story. If a massively parallel computer system is to be effectively used in demanding tasks from real live situations there are many other requirements.

For data intensive tasks, fast input and output are essential. The high internal bandwidth of the DAP (5.1 GBytes/second for the DAP 610 and 610C) allows fast input and output rates of 80 GBytes/second to be provided with no more than 3% impact on the compute capacity. AMT provides a range of custom-built fast I/O boards. These include an input/output computer that can under program control reformat the data during input or output. Disks with a storage capacity of up to 45 GBytes can be connected at transfer rates of 16 MBytes/second - this is about seven times faster than for normal systems.

One use of the fast I/O capability is to provide data visualisation. A high resolution, 1024 x 1024 pixels, colour screen can be driven at 60 Hz refresh rate from an AMT video output board plugged into the fast I/O of the DAP. This allows the user to "watch the computer at work" - it is a "window into the computation". This provides the user with instant feedback and is an enormous asset that costs only a few percent of the compute cycles.

For many tasks the most attractive features of the DAP are its small size, its low power consumption and the fact that it can be provided in ruggedized cases. This is particularly true for many applications where the ultimate objective is an embedded solution to a problem. For such users AMT offers a commercial system which is provided with user friendly tools that can be used for algorithm development. The same system can be used to prototype the application, since the DAP can be provided in ruggedized form. Lastly, AMT is willing to offer such users a technology licence to produce a customised embedded version but still within the same hardware and software environment. Thus AMT offers a single environment from algorithm development to a final embedded solution thereby reducing risks, cutting time scales and lowering costs.

The factors above, and many others that cannot be adequately covered in this short article, are often as important as sheer compute power. The excitement is that the AMT DAP/CP8 range offers a major advance in performance and price-performance but retains all the advantages of the earlier DAP range.

8. MARKET SIZE

The DAP is applicable in a wide range of market areas. It is customary for market analysts to divide the market into those relevant for different computer systems. The accumulative installed base for super computers is $4 billion, mini-supercomputers $0.4 billion, mainframes $140 billion, super-minicomputers $23 billion, general purpose workstations $1 billion, etc. (source: Electronic Trend Publications). SIMD systems can take a share of the above markets for new and existing applications. The current value of SIMD systems sold is no more than $40 million per year and hence the potential for growth is enormous.

9. FUTURE DEVELOPMENT SCOPE

Figure 3 also shows the scope for future DAP development. Systems can be improved by increasing the number of processors and/or by improving the power of the processors. Systems with a million times the power of a home computer system are possible and will be built. AMT has plans for further developments that will continue the enhancement of the DAP power. I have personally committed that we will increase the power of the initial DAP 510 by a factor of 1000 within a decade. In the first three years we will have achieved a factor of 40, which is well ahead of the target of a factor of 2 per year. I am confident that we will stay ahead.

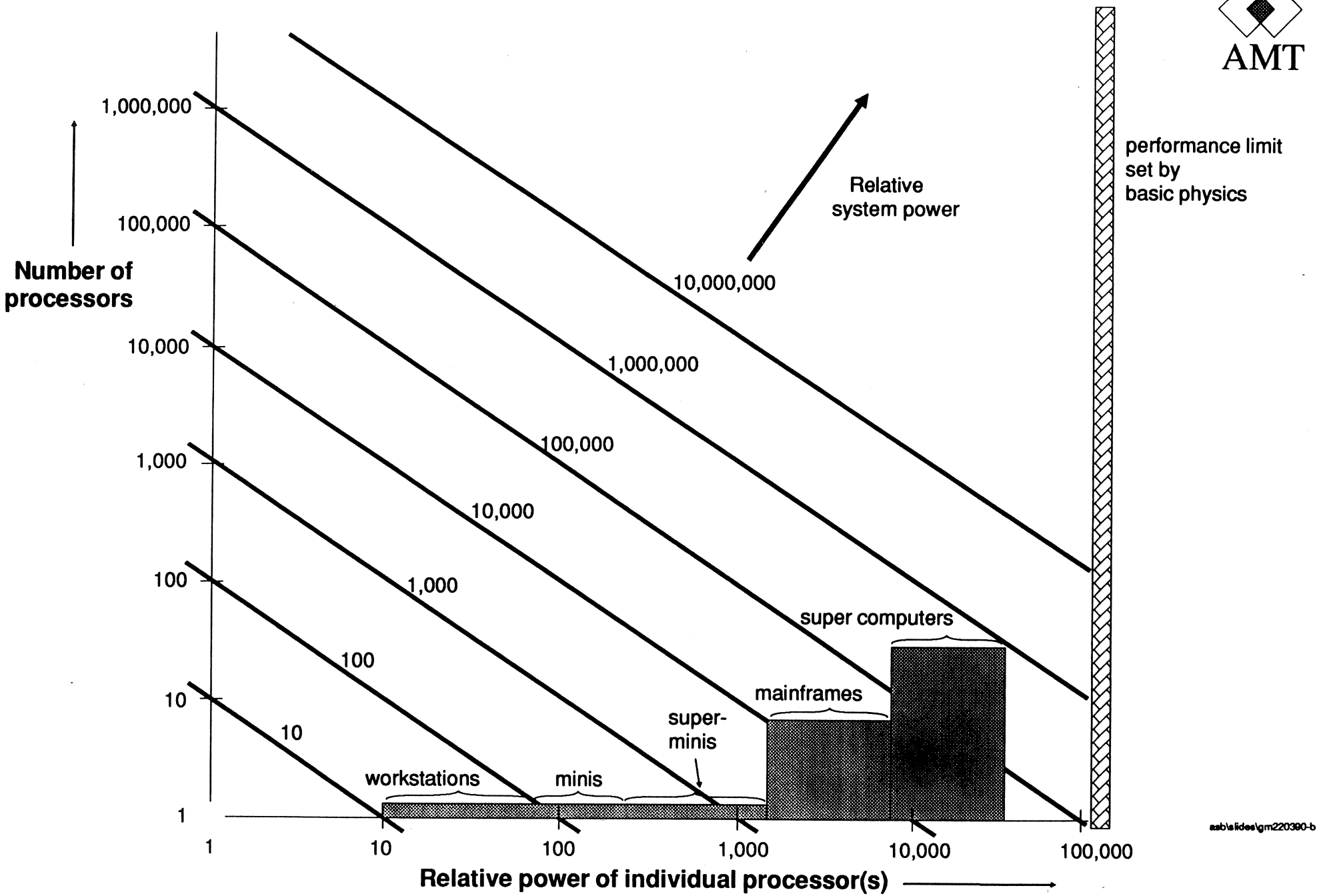


Figure 1 A plot of the number of processors against the power of an individual processor showing the overall system power resulting. The position of conventional computer systems is shown indicating how the most powerful computers are forced to become multi-processor systems (ie parallel systems) as the uni-processors approach the physical limits of what can be achieved.

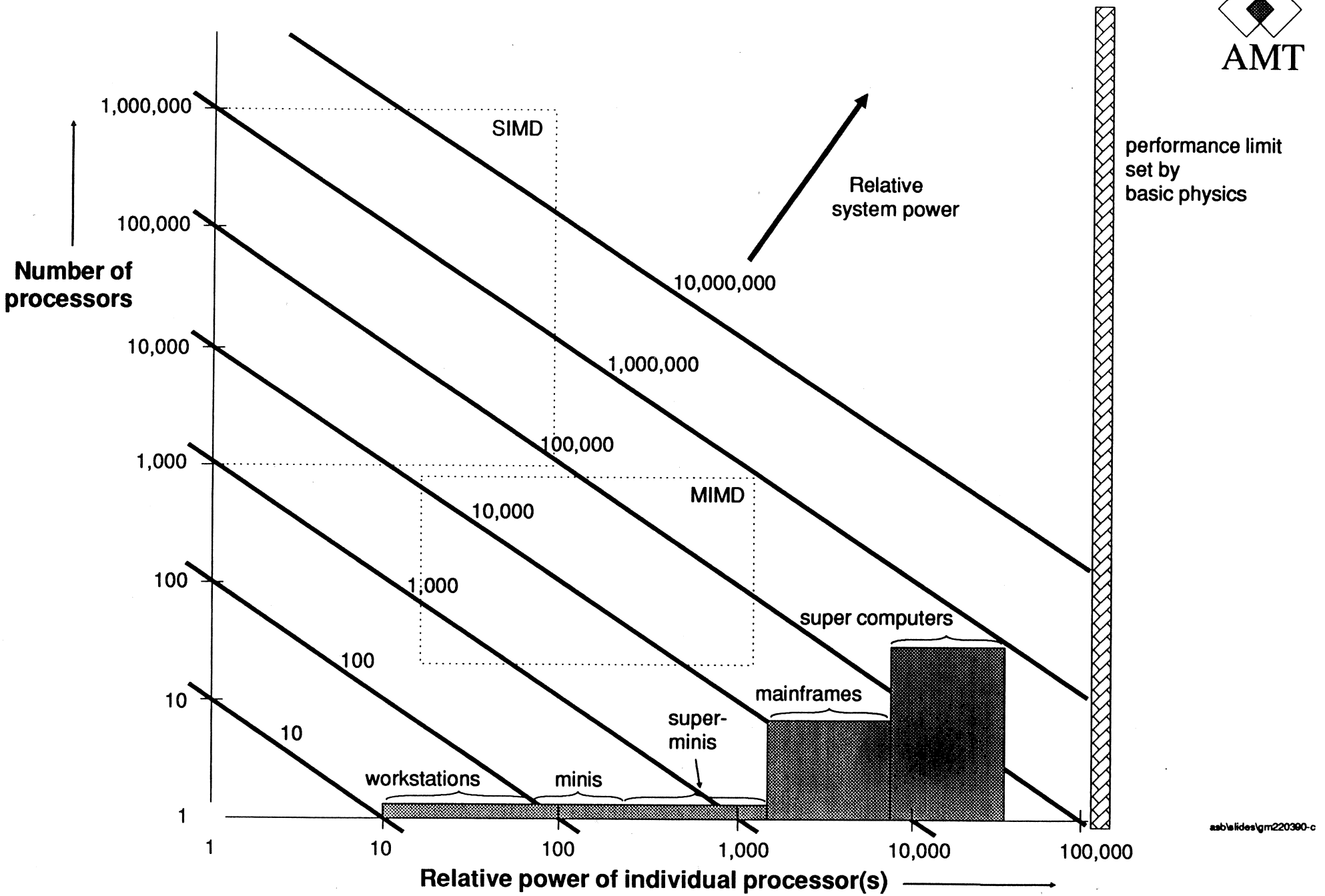


Figure 2 The domains covered by current and future SIMD and MIMD systems are shown. SIMD uses very large numbers of simple processors and MIMD much smaller numbers of more powerful processors. Both forms of system can provide super-computer performance.

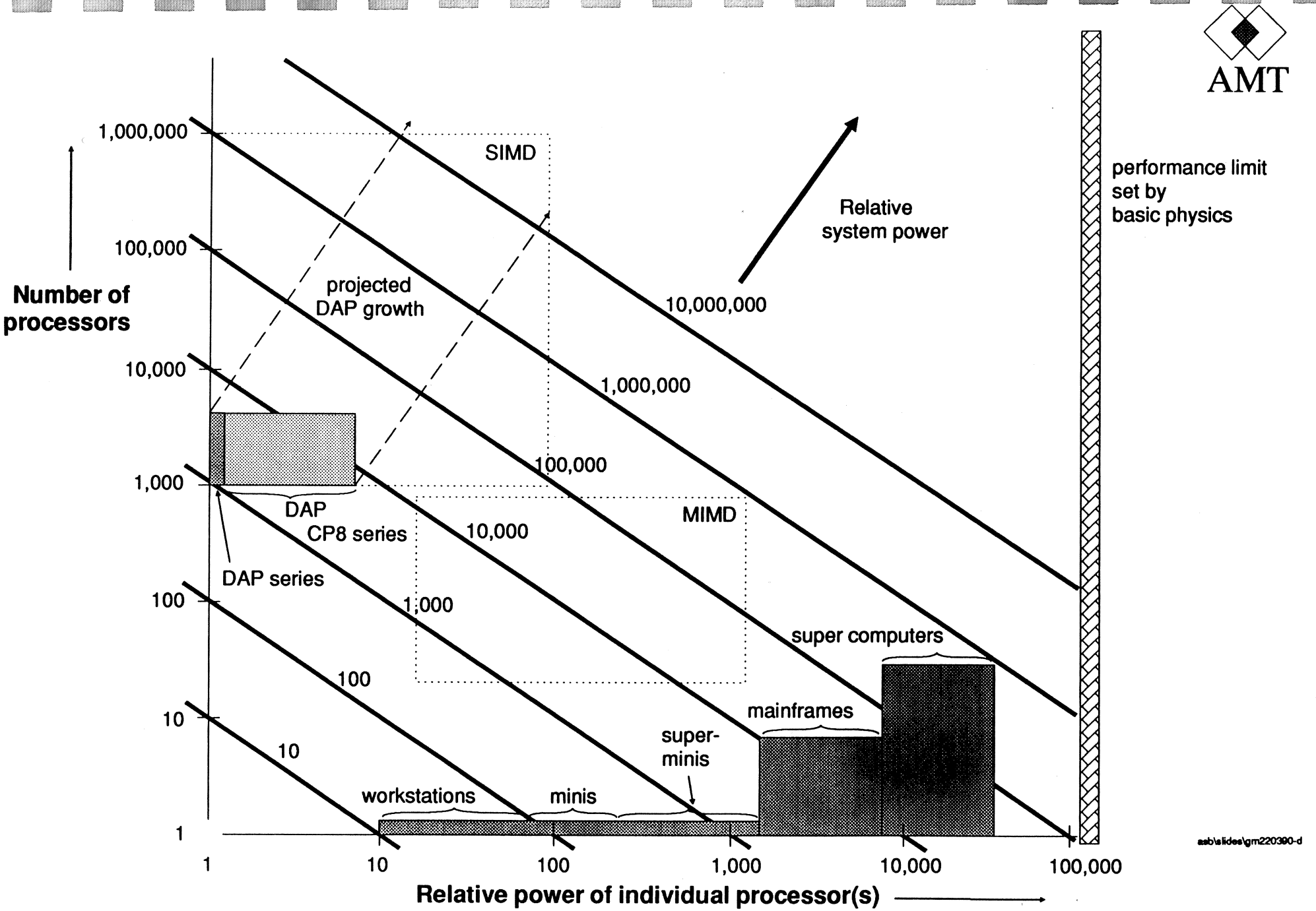


Figure 3 The figure shows the position of the current DAP range and that of the new DAP/CP8 range. The 610C model has a performance in the super computer range. The diagram shows the DAP growth path towards much higher performance systems.

1 Introduction

About CPP

Cambridge Parallel Processing (CPP) is a privately held, American-owned company, and is a leading international supplier of advanced, fine grained, massively parallel computer systems.

CPP's Product Line

CPP's product line comprises many different configurations of the Gamma II^{Plus} series of computers. This series offers:

- 1024 and 4096 processor variations
- exceptional price/performance ratio — both for workstations and as network servers
- stand-alone computer system configuration for dedicated applications
- power of up to 2.4 GFLOPS (32-bit) and 123 GOPS (8-bit integer addition)

Based on over twenty years of proven technology, the Gamma II^{Plus} incorporates the fourth generation of CPP's proprietary DAP (Distributed Array of Processors) technology, and employs an SIMD (Single Instruction Multiple Data) architecture.

Application Areas for the Gamma II^{Plus}

The SIMD architecture makes the Gamma II^{Plus} ideal for image and signal processing. It also provides a natural and efficient solution to many problems in banking, insurance, transportation, engineering, medicine, manufacturing and communications — in fact, any application where large data sets are subject to multiple operations.

Application areas include:

- Image Processing
- C^3 (Command, Communication and Control)
- Signal Processing
- Data Compression
- Data Mining
- Statistical Analysis
- Neural Networks
- Optical Character Recognition
- Bio-Sequencing
- Free Text Database Searching
- Medical Imaging
- Computational Fluid Dynamics
- Robotic Vision
- Dynamic Linear Programming
- 3D Graphics
- Minefield Detection

Applications Support

CPP provides an extensive range of powerful software development tools and application support libraries to ensure all software development on the Gamma II^{Plus} fully exploits the system's exceptional power and versatility.

Application developers can select either C++ or Fortran-Plus (Fortran 77 with parallel processing extensions).

For More Information

If you would like further information, please contact one of the CPP offices shown below:

Worldwide Headquarters
16841 Armstrong Avenue
Irvine, CA 92606
USA

Tel: (949) 724 8300
Fax: (949) 724 8399
email: info@cppus.com

European Headquarters
Centennial Court
Easthampstead Road
Bracknell, Berks RG12 1YQ
UK
Tel: (01344) 861024
Fax: (01344) 305544
email: info@cppuk.co.uk

Overview

Main Components

The Gamma II^{Plus} has four main components:

- Processor Elements (PEs) with array memory.
- Master Control Unit (MCU) with code store.
- VMEbus with microprocessor controller, host interface (optional) and VME slots.
- Fast Input/Output Controllers (optional).

The relationship between these components is displayed on the following diagram:

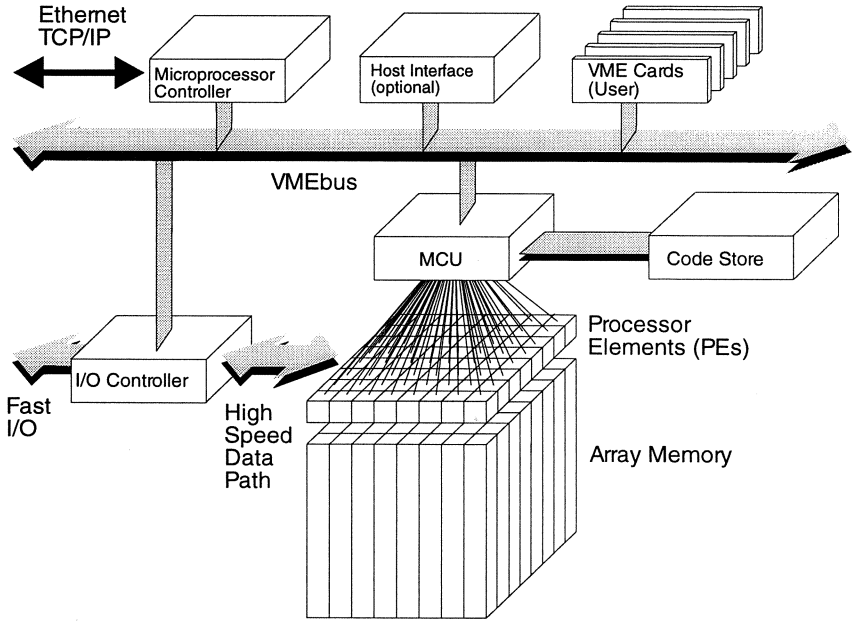


Figure 2.1 *The Main Components of the Gamma II^{Plus}*

Processor Elements

The computational core of the Gamma II^{Plus} is an array of Processor Elements (PEs). The Gamma II^{Plus} achieves its performance through the power of its PEs and their high connectivity.

Master Control Unit

Central control of the Gamma II^{Plus} array is vested in the MCU, which includes a control processor and code store holding DAP instructions. The MCU communicates with the processor array via dedicated address, data and control paths. The PEs operate synchronously under control of the MCU.

VMEbus

The VMEbus interconnects the main components of the system, and provides a link to an optional remote host workstation. It is used for both data transfer and control messages, and may have standard VMEbus cards connected to it.

FIO Controllers

The Fast Input/Output Controllers connect external devices to a high speed data path and so to the array memory of the Gamma II^{Plus}.

Processor Elements

Each PE in the array contains:

- a 1-bit processor
- an 8-bit processor
- array memory

The 1-bit and 8-bit processors work together, enabling each PE to perform tasks ranging from single-bit operations through to multi-byte floating point arithmetic.

Connections between PEs

The PEs are connected to each other in a two-dimensional array; for current Gamma II^{Plus} models this can be either a 64 × 64 array (Gamma II^{Plus} 4000) or a 32 × 32 array (Gamma II^{Plus} 1000). (See also "PE Connections" on page 2-5.)

Processor Selection

The Gamma II^{Plus} compilers select which processor to use in a way that is entirely transparent to the user.

1-Bit Processors

Each 1-bit processor (Figure 2.2) has three 1-bit registers called Q, C and A. Each of these registers is used for a variety of purposes.

Q and C Registers

It is convenient to think of the Q register as being an accumulator and the C register as a carry register. Arithmetic and logical functions are performed within the PE by the adder, which adds the contents of the Q register, the C register and a third single-bit input (typically a bit read from array memory), producing a sum and a carry-out. Addition of multiple-bit data is achieved by a sequence of such additions operating on successive data bits, starting at the least significant bit.

A Register

The main purpose of the A register is to provide *activity control*. Some instructions that write a result from the 1-bit processor to array memory are conditional on the value of the A register, thus permitting data to be written or not depending on some local condition.

Q-A-C Register Planes

As all of the PEs run in lock-step, it is convenient to think of the Q, C, and A registers of every 1-bit processor as forming three *register planes*: the Q-plane, the C-plane and the A-plane. These are shown in the 1-bit processor array in the upper part of Figure 2.3 on page 2-6. (Other parts of the figure show the array memory and 8-bit processor array, which are discussed later.)

D Register

The D register shown in Figure 2.2 is used for input or output under control of the Fast Input/Output unit (page 2-18). Data to be output is transferred one plane at a time out of the array memory to the D-plane. The data is then transferred from the D-plane through the

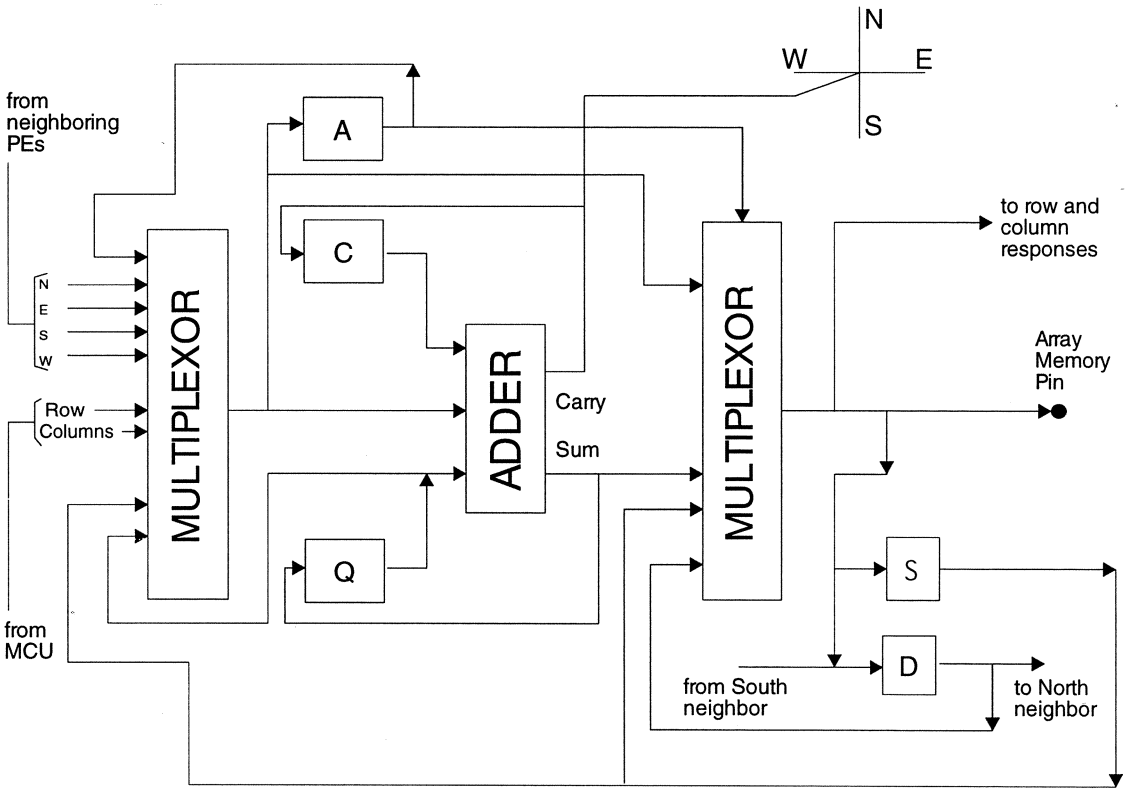


Figure 2.2 *1-bit Processor*

edge of the PE array to the external device, independently of any other processing. Data is input by reversing this procedure.

S Register

The S register is used by instructions that both read from and write to the array memory.

PE Connections

Neighbor Connections

Each PE is connected to four neighbors; north, south, east and west, as shown in Figure 2.4. Using these connections, data can be shifted from the Q register of each 1-bit processor to the Q register of its neighbor in any one of the four directions. PEs at an edge of the array are connected to those at the opposite edge of the array, so that shifts can "wrap-around" if required.

Row and Column Data Paths

PEs are also connected to row and column data paths (Figure 2.4). The data in an MCU register can be broadcast along these paths so that the bit n of the register is received by all the PEs in row n (or column) of the array. In this way a vector of bits can be copied to

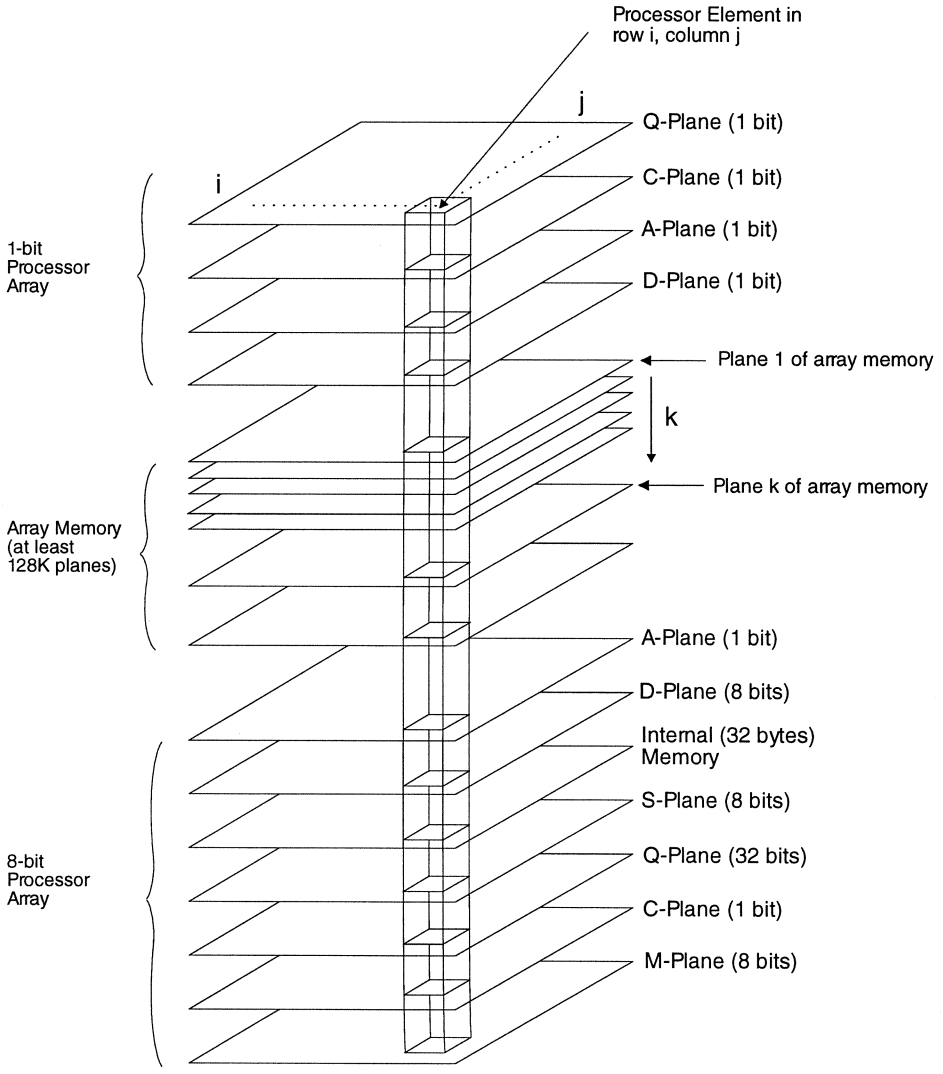


Figure 2.3 *Processor Element Array*

every row (or column) in one instruction. The row and column data paths can also be used to extract a row (or column) and place it in an MCU register or to AND together all the rows (or columns) and place the result in an MCU register.

Assembly Language for 1-Bit Processors

See also "Assembly Language for 8-Bit Processors" on page 2-9.

The 1-bit processors execute the Array Processor Assembly Language (APAL) instructions broadcast to the array by the MCU. APAL also includes scalar, control, and loop control instructions which are executed solely by the MCU. The high-level language

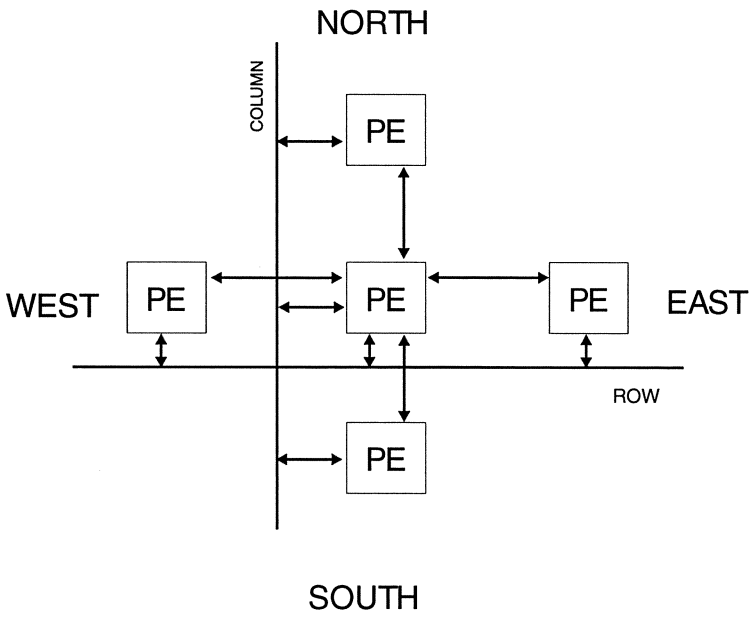


Figure 2.4 *PE Connections*

program (C++ or Fortran-Plus) is compiled into a series of calls to intrinsic functions that are linked into the program. A typical intrinsic might shift a user array by loading each plane of memory in turn into the Q-plane, shifting Q by the required distance, and storing it in the corresponding result plane.

8-Bit Processor

Figure 2.5 on page 2-8 shows the main components of an 8-bit processor. Most of the registers and data paths are 8 bits wide.

Register Names

Many of the register names are the same as those in the 1-bit processors, and, as with the 1-bit processors, can be thought of as forming planes (bottom of Figure 2.3 on page 2-6). However, the 8-bit processors' registers are distinct from the 1-bit processors' registers

Internal Memory

Each 8-bit processor has 32 bytes of internal memory which is used to hold the operands, results and intermediate values of 8-bit processor operations.

D Register

The D register, at the bottom right of Figure 2.5, is used to transfer data to or from the array memory (using a 1-bit data path), and to and from internal memory (using an 8-bit data path).

A Register

The A register shown in Figure 2.5 is a copy of the A register associated with the 1-bit PE, and likewise provides activity control.

Some instructions that write a result from the 8-bit processor to array memory are conditional on the value of the A register, thus permitting data to be written or not, dependent on some local condition.

S Register

The S register is an 8-bit operand register, mainly used when reading data from internal memory.

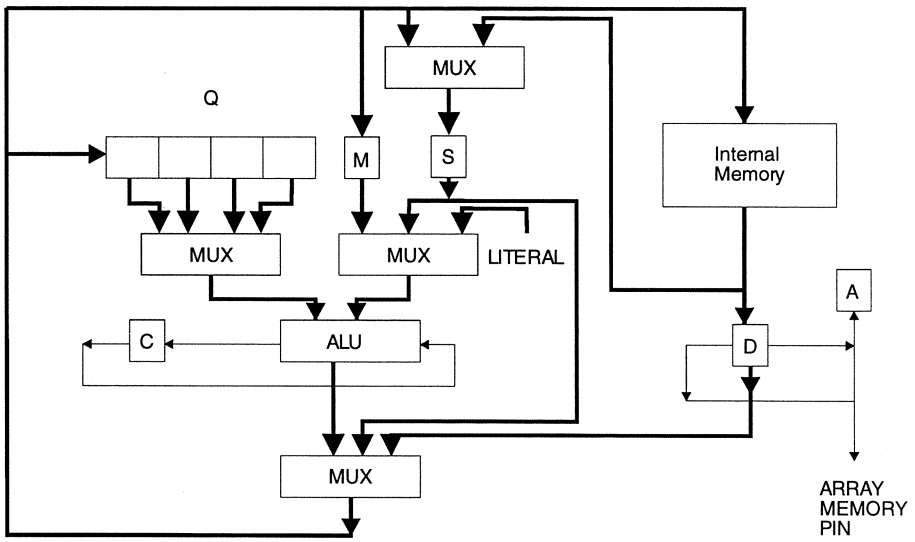


Figure 2.5 *The 8-bit Processor*

Q Register

The Q register acts as an accumulator and is 32 bits wide. Any one of its four bytes can be selected for use as an operand of the 8-bit Arithmetic-Logical-Unit (ALU). The complete Q register can be shifted right by one bit or one byte, and in the latter case the 8-bit value shifted in comes from the ALU via the multiplexer shown below it. Shifting of Q can be done unconditionally or can be dependent on a local condition.

M Register

The 8-bit M register may be shifted left or right. Its least significant bit may control the multiplexer on the ALU output or control the shifting of the Q register. During a multiply the M register holds a byte of the multiplier.

Multiplexer

The multiplexer shown below the ALU in Figure 2.5 allows selection of either the ALU output or the contents of the S register, thus giving a form of conditional control.

Arithmetic Logic Unit

The ALU provides 8-bit addition and subtraction operations, and a variety of bit-by-bit logical operations.The carry out from arithmetic operations can be clocked into the 1-bit C register and used as a

carry in for a following instruction, thus allowing multiple-byte operations to be implemented.

Performance

A Gamma II^{Plus} 4000, running at 30 MHz can perform 4,096 8-bit operations every 33 nanoseconds giving a total rate of 120 GOPS.

Assembly Language for 8-Bit Processors

See also "Assembly Language for 1-Bit Processors" on page 2-6.

The 8-bit processors execute Coprocessor Assembly Language (CPAL) instructions broadcast to the array by the Master Control Unit (MCU). CPAL is used only for intrinsic functions such as floating point arithmetic on arrays.

CPAL Intrinsic Functions Library

A CPP-supplied library of such intrinsics — performing a variety of functions on different types and lengths of data — is loaded automatically at system initialization. The high-level language (Fortran-Plus or C++) compilation system generates calls to these CPAL intrinsics, and each application program is linked to the required intrinsics as it is loaded.

There is no mechanism to call CPAL direct from Fortran-Plus, but CPAL can be called from Fortran-Plus by means of an intermediate APAL routine. Such CPAL code is loaded and linked at the time the APAL routine is loaded, as a part of the complete program.

APAL Control of CPAL

Each CPAL intrinsic comprises a sequence of low-level instructions which, once started, run to completion without looping or branching. All control of the CPAL instruction stream is performed by APAL instructions; these initiate the execution of CPAL intrinsics (with task queueing to a depth of one), and can test for the completion of those intrinsics. Once execution of a CPAL intrinsic has been started, it executes autonomously and in parallel with the execution of subsequent APAL instructions.

Data Transfer

The APAL instruction stream is also responsible for transfer of data between the array memory and the internal memory of the 8-bit processors. At the array memory these data transfers occur one bit-plane at a time. Within the 8-bit processor the data is buffered within the 8-bit D register shown in Figure 2.5, and at every eighth data bit, a complete byte is written to, or read from, the internal memory. If the 8-bit processors are executing CPAL at that time, then a clock cycle is stolen to access the internal memory.

CPAL Pipelining

Usually the compilation system allocates the internal memory in such a way that while a given CPAL intrinsic is being executed, the result of a previous intrinsic is written to array memory, and the operands for the next intrinsic are copied from array memory to internal memory. By this means, an efficient pipeline of operations

is established, making good use of both the processing capability and the bandwidth of array memory.

Array Memory

PE Memory Size

Each PE has a one-bit wide, direct connection to its own section of the array memory. The size of each PE's section of array memory depends on the configuration of Gamma II^{Plus}but is in the range 128 Kbits to 1 Mbit.

Array Memory Shape

The array memory can be regarded as a three-dimensional array of bits consisting of at least 128K *memory planes* (Figure 2.3 on page 2-6). A memory plane consists of the bits at the same address in each PE's memory. Bit (*i, j*) of a memory plane is therefore associated with PE (*i, j*).

The array memory can also be regarded as a sequence of *rows*, the length of a row being determined by the size of the Gamma II^{Plus} PE array. On a Gamma II^{Plus} 1000 machine each memory plane can be regarded as 32 rows of 32 bits each and on the 4000 as 64 rows of 64 bits. In general, each row comprises a number of 32-bit words. On a Gamma II^{Plus} 1000, words and rows are equivalent. On a Gamma II^{Plus} 4000, a row is two words.

Data Representation

The Gamma II^{Plus} is not committed to any particular representation of data and generally regards data as arrays of bits, the interpretation of which (as fixed or floating-point numbers, for example) is defined by the software.

There are many possible mappings, or arrangements of data, on the Gamma II^{Plus}. There are, however, two simple, natural mappings:

- **Horizontal Mode**, in which successive bits of a data item are mapped onto successive bits of a single row of a memory plane.
- **Vertical Mode**, in which successive bits of a data item are mapped onto the same bit position in successive memory planes.

These mappings correspond to the scalar (horizontal) and the vector or matrix (vertical) storage modes of the Gamma II^{Plus} high-level programming languages.

Memory Performance

All processors can simultaneously transfer one bit to or from memory within a single Gamma II^{Plus} clock cycle; this means that a Gamma II^{Plus} 4000, running at 30 MHz, has a transfer rate of 120 Gbits/sec.

Master Control Unit

The Master Control Unit (MCU) is a dedicated pipelined processor whose role is the overall control of the Gamma II^{Plus} array. It includes scalar processing facilities, but is mainly concerned with issuing instruction streams to the array of 1-bit processors and the array of 8-bit processors.

The main components of the MCU are shown in Figure 2.6 on page 2-12, and the functions of each part are outlined in the following sections.

APAL Control of MCU

The MCU is controlled by the APAL instruction stream. Some of these instructions it executes itself as control operations or scalar arithmetic, but most of the instructions are merely decoded and passed to the array for execution by the 1-bit processors.

APAL Code Store

APAL instructions are 32 bits wide and are held in a dedicated code store. This is organized in banks of 1Mbyte, and the MCU may be populated with one, two, or four such banks, giving a maximum configuration of 4Mbyte, or 1M instructions.

Master Control Chip

At the heart of the Master Control Unit (MCU) is the Master Control Chip (MCC), which interfaces to the rest of the system via buses carrying address, data, and decoded array instructions.

The MCU fetches and decodes APAL instructions from the Code Store. A high proportion of APAL instructions executed take one clock cycle, but for those that take longer, a lower level of microcode control is used. The microcode store is also used to decode part of the array instruction, even for single-cycle instructions.

DO Loop

As well as the usual jump instructions, the MCC includes logic dedicated to an APAL DO instruction. This provides efficient looping over a sequence of instructions, such as those used to implement operations on each bit-plane of a data set in turn.

Master Control Chip Registers

Within the MCC, a triple port register file holds 14 general purpose 32-bit wide MCU registers, plus a further register accessible only by privileged (system) code. These registers may hold scalar data or the addresses of array data, and arithmetic, logical, or shift operations may be applied to them. MCU registers may be broadcast to the array, or they may be loaded with data returned from the array.

Array Memory Addresses

The MCC generates addresses for the array memory. Often a complete bit-plane is addressed, but other instructions provide for accessing a row, column or word within a plane. Addresses may be

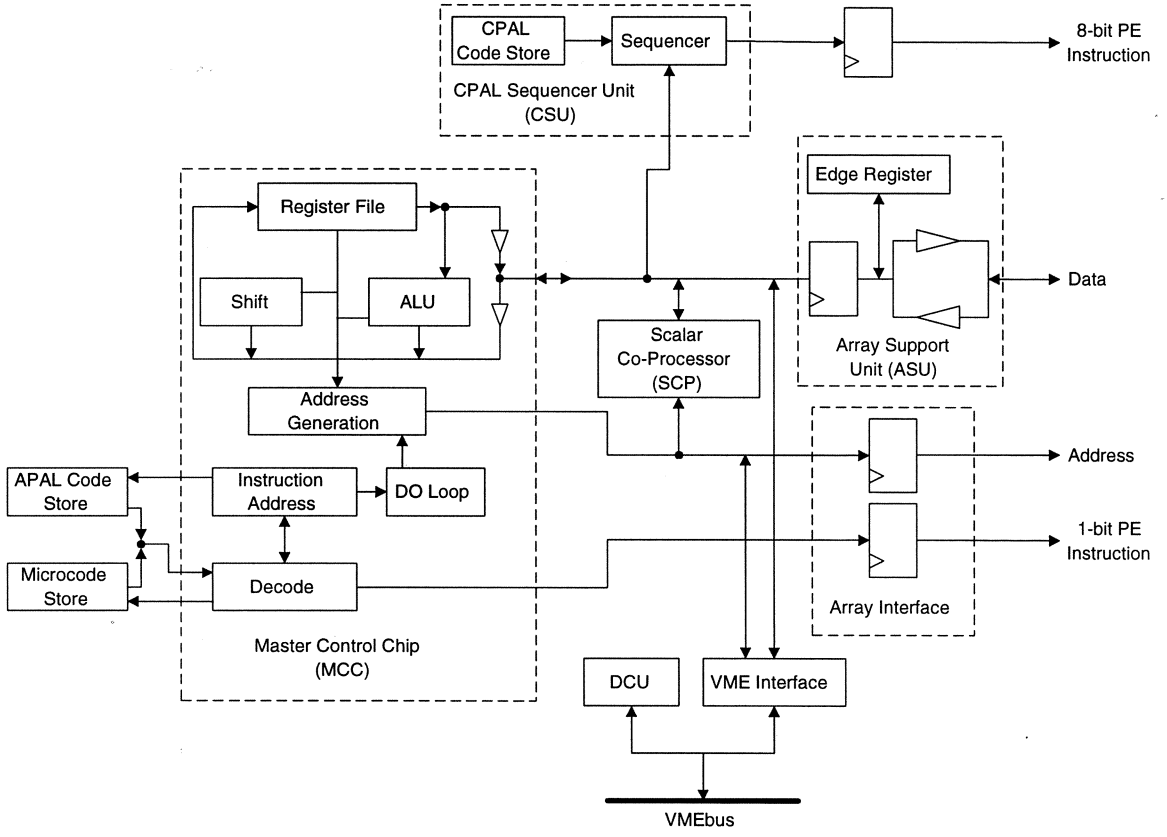


Figure 2.6 *The Master Control Unit (MCU).*

modified by the contents of one of the MCU registers, and may be stepped (incremented or decremented) on each pass of a DO loop, thus providing efficient access to successive planes, rows, or words within array memory.

Scalar Coprocessor

An integer scalar multiplier chip is connected as a coprocessor to the MCC. This extends the range of scalar operations that may be applied to MCU registers.

Array Support Unit

The Array Support Unit (ASU) acts as an interface between the data bus within the MCU, which is 32 bits wide, and the data bus in the array, whose width matches the edge size of the array. For a 64 × 64 array, the array data bus is 64 bits wide, for example. When sending data to the array, the MCU data may either be replicated or zero extended (if necessary) to fill the array bus. When receiving data from the array, the MCU may (if necessary) select one of the words from the array data bus.

ASU Edge Register

Within the ASU there is a register known as the Edge Register, whose size matches the array edge size. It may thus be broadcast to the array, or its contents loaded direct from the array data bus, regardless of the size of the array. The Edge Register is addressed by the MCU in the same way as the MCU registers. It can be shifted and tested, but cannot otherwise take part in MCU scalar operations.

CPAL Sequencer Unit

Instructions for the 8-bit processors are issued by the CPAL Sequencer Unit. CPAL instructions are held in their own code store which can hold 65536 instructions of 32 bits.

APAL Initiates CPAL Execution

APAL instructions are used to initiate the execution of CPAL code sequences, and to transfer data between array memory and the internal memory of the 8-bit processors. Once a CPAL intrinsic has been started, the CSU continues to fetch CPAL instructions and issue them to the 8-bit processors independent of the APAL instruction stream, until the end of the CPAL intrinsic is reached. Start commands for CPAL intrinsics may be queued to a depth of one, and when one intrinsic ends another may be started automatically from the queue.

APAL instructions can test the state of the CSU to determine whether it is idle, busy, or busy with another intrinsic in the queue.

Array Interface

The array address and the decoded instructions for the 1-bit and 8-bit processors are buffered and distributed by array interface logic. The equivalent function for the array data is implemented by the ASU.

VMEbus Interfaces

The MCU provides both master and slave interfaces to the VMEbus. This allows devices attached to the VMEbus to access data in the array memory, and to load the APAL and CPAL code stores and the microcode store at system initialization, and when loading individual application programs.

Incoming Requests from VMEbus

An incoming request from the VMEbus causes the APAL instruction stream to be suspended while the required memory access is performed under control of the MCC microcode.

Privileged APAL Code

Privileged APAL code can access system resources via the VMEbus, as well as code store and memory-mapped control locations within the Gamma II^{Plus} itself.

Diagnostic Control Unit

The Diagnostic Control Unit (DCU) provides control for the hardware initialization sequence of the MCU and array. It connects to the VMEbus as a slave interface and is thus controlled from that bus.

Original : RNI
Copy : SM ✓

Profs Wallace, Tobert ✓
Dr Sutton, Williams
Dr John Collins.

DAP work in Edinburgh

It may soon be pressing to know whether there is a perceived future for DAP work in Edinburgh in the broader context. At present we have two DAPs in the University, both in the JCMB. The first (icarus) has been supported by an Alvey grant which terminates 30.9.89, and the second (canopus) is Computer Board supported until 1991 and the CB requirement was that it should be operated by EUCS.

I had gone ahead with the Alvey project, which we had initiated in the early months of the whole Alvey programme, in the hope that this would in some measure sustain the wide community of DAP users that had been served by the two ICL DAPs; the work that we are contracted to deliver under this project is not work that I would normally choose to do as my research. The community has dwindled, I think for three reasons. The first is that there was much more of a delay in producing the first machine than was anticipated, the second is that the machine, as hosted on a SUN, is not in a good multi-user environment (but Dr Trew's work on the batch scheduler has alleviated this to some extent) and did not have sufficient resource (now partly alleviated by the acquisition of a bigger disc drive), and the third is that the compute resource, equivalent of about 20 T800s, is insufficient for the competition.

There have been attempts to boost the resource; the URC in Molecular Science would have had this effect, and so too would either of two applications which went in when that short-listed URC failed. There was also an application to the DTI at the time when DTI were instructed to spend no more new money, and this has in turn led to a double IED proposal, that part of which would have boosted the DAP being rejected. Our big IRC effort is another failed support mechanism, and a small attempt at a partial upgrade of icarus under the Molecular Electronics Initiative has just failed alpha. When the Alvey grant expires, there will be little funded work to cover the maintenance on icarus, let alone an upgrade. We made the mistake of going for a second DAP 510 rather than an upgrade to one DAP 610, but this decision was taken when there were reasons for optimism with other grant applications.

The future therefore seems bleak, and this is supported by the rumour gleaned at the last ECS site visit that SERC would not support DAP work in Edinburgh while supporting the ECS. I get the impression that a viable programme of DAP work needs considerable input from more than me, especially important being Computer Science input. Although I consider the perceived SERC policy as blind, I do not discount it, and have written to SERC for clarification!

I refer above to a double IED bid. This was for making a fast interface between DAP and ECS, and for proving this heteroarchitecture by mounting some specific applications after installing a DAP 610. The interface work seems to have been approved, but the applications plus DAP 610 have been rejected. There might be half a chance to try for a second bite at IED, but even without such support there is interesting work that could be done on the joint machine. The question which will arise is whether to accept the IED project for the interface, as this does involve Edinburgh marginally, or simply to admit that the DAP work in Edinburgh is a thing of the past and reject the IED offer when it comes. If there is no interest, and it is left to me to make a lone decision, I will not go it alone.

This is a difficult decision. The DAP work has got us to where we stand at the moment with the ECS, but we should not take decisions for sentimental reasons. Nevertheless, I need to know what commitment there is in the computer user community in Edinburgh so that I can organise my effort to some purpose.

G.S.Pawley

22.3.89

RECEIVED
28 MAR 1989

ACTIVE MEMORY TECHNOLOGY

AMT

Active Memory Technology Ltd.
65 Suttons Park Avenue
Reading RG6 1AZ
United Kingdom
Telephone: 0734 661111
Telex: 846931
Fax: 0734 351395

Dr R N Ibbett
Computer Science Department
University of Edinburgh
King's Buildings
Mayfields Road
Edinburgh
Scotland
EH9 3JR

RECEIVED
- 8 MAY 1989

2nd May 1989

Dear Dr Ibbett,

COMPUTER BOARD HIGH PERFORMANCE INITIATIVE - 1989

You may recall that last year, the Computer Board introduced an initiative to promote high performance parallel computing. AMT were successful in securing six orders from Universities funded by that initiative which emphasised the interest shown in the DAP by the academic community. Edinburgh University has two DAP-510 systems, one in Molecular Biology and one in Physics.

We are now led to believe that following a major over-subscription last year under the initial scheme, the Board has obtained increased funds for further rounds of the initiative in 1989, the announcement being due soon. I enclose a copy of the 1988 Computer board letter for your reference.

The DAP offers very impressive performance in numerically intensive computations, particularly in variable wordlength application areas. We have identified three areas where this has proved most significant:

Signal and Image Processing
Simulation and Modelling
Text/Character matching and retrieval

To assist you with the preparation of a potential application, I enclose a general quotation which gives you the costs and configuration details of the DAP systems. I also enclose a document which catalogues many of the papers that have been published about use of the DAP architecture in a variety of application areas and hence may be of some help to you and your potential users. I would be pleased to provide copies of relevant papers on request.

I will contact you in the near future to discuss any ideas you may have which could result in a submission for funding for a DAP system. In the mean time, should you need any further information, please give me a call.

Yours sincerely,

Ian Mitchell
Sales Manager

Registered in England No. 2012221

The Computer Board for
Universities and Research Councils
Elizabeth House York Road London SE1 7PH
Direct Line 01-934 9854
Switchboard 01-934 9000
GTN Number 2914
Telex 23171 DESLON G

To all Vice Chancellors
and Principals

Your reference

Our reference

23 March 1988

INITIATIVE ON HIGH PERFORMANCE DISTRIBUTED FACILITIES

As indicated in my letter of 8 February the Board has decided to use part of the recent increase in its resources to fund an initiative on high performance distributed facilities. A total capital sum of £1 million has been allocated and bids are invited for grants to support the purchase and running of such systems. The Technical Options Group of the Joint Policy Committee on National Facilities for Advanced Research Computing is advising the Board on the programme.

2. The aim of the initiative is to improve the availability of small, novel architecture systems (costing about £100,000 to £200,000) of the types which are felt most likely to have significant potential for advanced research computing. The Technical Options Group has identified three types of equipment which it feels would be worth supporting in this way. They are :-

a. Coarse-grain, shared memory, multiple instruction/multiple data (MIMD): (eg Alliant FX/8, Encore Multimax, Sequent Balance and Symmetry);

b. Coarse-grain, distributed memory, MIMD: (eg Meiko Computer Surface, Intel iPSC/2); and

c. Fine-grain, distributed memory, single instruction/multiple data (SIMD): (eg AMT DAP, Thinking Machines CM-2).

3. The maximum capital grant will be £200,000 and recurrent costs will be supported for a period of 5 years, with an annual limit of 10% of the capital grant awarded. In order to provide as many systems as possible and to distribute them as widely as possible the Board has agreed that the following factors will be taken into account when evaluating bids:

a. university and/or other contributions to the costs;

b. local skills and expertise;

c. adequate communications to JANET/JNT standards;

d. evidence of interests from a group or consortium prepared to share facilities;

e. geographical location; and

f. involvement of UK suppliers.

4. Projects must be managed by the computer centre at the lead university and a progress report will be required for each of the first three years. The Board will reserve the right to redeploy the equipment if, in its view, the reports are unsatisfactory. The Board may be prepared to fund a support post at the university for the first three years of the project.

5. Universities are invited to submit bids for support under this initiative. Proposals should occupy not more than four sides of A4 paper. Brief details should be given of the type and supplier of equipment proposed (including an equipment list) and of the applicability of the evaluation factors listed above.

6. Twenty copies of proposals should be sent to the Computer Board Secretariat, Room 5/99, at the above address. The first round of evaluation will be at a meeting of the Technical Options Group on 27 May 1988 and applications should reach the Secretariat by 6 May 1988 for consideration at this meeting.

Yours faithfully

E J Herbert

E J Herbert
Secretary

cc Directors of Computer Centres
University Grants Committee
Science and Engineering Research Council

SAMPLE COSTS and CONFIGURATIONS

Reference: IM/Q1183/D
2nd May 1989

Item	Product	List Price	Discount	Net Price
1.	DAP 510-4	£87,600	25%	£65,700
2.	Video Output Board	5,500	25%	4,125
3.	Hi-Res Monitor	2,600	-	2,600
4.	Run Time Software APAL Signal Processing Library Image Processing Library	)		£5,000
5.	Fortran Plus General Support Library DAP Simulator	)		Nil
6.	Maintenance per annum			10% of price

Options

1.	VME Subsystem	£ 4,100	25%	£ 3,075
2.	MICROVAX - DR11W Interface (requires Option 1 VME Subsystem)	2,500	15%	2,125
3.	SUN SCSI Workstation Interface			Nil
4.	SUN File Server Interface	3,000	15%	2,550

Other Options

Incremental Costs (before application of academic discount)

1.	to 510-8	£ 26,600
2.	to 510-16	69,400
3.	to 610-16	144,400
4.	to 610-32	237,400
5.	Additional 1/2 Mb code store	3,700

Notes:

1. Option 2 assumes a MICROVAX host connection to a customer supplied DRV11 interface. The price for the host is not included in the above quotation.

2. Item 5 is supplied by CHEST for an annual support payment to CHEST of £500.00 for minimum period of 3 years.

3. All prices exclude VAT.

ACTIVE MEMORY TECHNOLOGY

DATABASE OF

DAP REFERENCES

April 1989

Copyright 1989 by Active Memory Technology
No part of this publication may be reproduced
in any form without written permission from
Active Memory Technology.

Active Memory Technology Ltd
65 Suttons Park Avenue
Reading RG6 1AZ , United Kingdom

Active Memory Technology Inc
16802 Aston Street , Suite 103
Irvine , California 92714 , U.S.A.

ACTIVE MEMORY TECHNOLOGY

DATABASE OF DAP REFERENCES

Introduction

The AMT Library at Reading contains a collection of DAP-related references. These have been collected from various sources: some are published in journals, some in conference proceedings or in books, some in internal or unpublished reports.

The DAP references are identified by Author, Year, Title, Source, an AMT reference number, and some keywords. The reference details have been entered into a PC software package called PAPERBASE, which allows the selection of references by "keywords". For example, you can select on "neural", or "performance", etc, or on any word in the Titles. The package will list the references you require and indicate where a copy is located. The database will give you either of the following locations:

(a) an AMT reference number - indicating that a copy is available in the AMT Library at Reading (eg UK7.16)
(b) an AMT Technical Report number (eg AMT TR6)
(c) an AMT Applications Note number (eg AMT AN7)
(d) an ICL Report number (eg ICL CM23)
(e) or where it is published.

Copies of the AMT TR, AMT AN or ICL CM Reports are available from the "Central Documentation" cupboard at AMT, Reading.

Full List of DAP References

A full list of DAP references (in alphabetical order by the first author) contained in the AMT database is given at the end of this report. New references are being added from time to time and you should request a current list at the time of your search.

Keyword Search

You can make a keyword search from within the Titles of the references and from a standard set of keywords which have been entered as part of the input reference data. A list of the current standard keywords is given in Table 1, although new keywords are being added from time to time.

Input File Format for New References

Each reference occupies four lines in a file. The first line gives the author(s) and year:

Smith, AB, Jones, CD, 1989

The second line is the Title, which can be as long as you wish.

The third line can have a number of forms, depending upon whether the entry is a published journal paper, a report, a conference paper or within a book:

(a) For a journal, specify:
Parallel Computing,4,259,548,keyword,keyword
This gives the journal title, volume number, starting and finishing page and keyword (one or more, each separated by a comma).

(b) For a report, specify:
report,Edinburgh University,UK2.1,,,keywords
The third item indicates where a copy of the reference is located.

(c) For a conference paper, specify:
proc,conference name,\\,organisation,place,,,keywords

(d) For a book, specify:
in,title,\\,Ed,publisher,place,,,,keywords

The fourth line must begin with a * and gives the location of the reference. It can be in any form, such as:

* UK3.7 for an AMT reference number
* TR3 for an AMT Technical Report number
* AN6 for an AMT Applications Note number
* CM12 for an ICL Report number.

Access to the Database

It would be better for an expert to access the database to carry out a keyword search on your behalf, enter new references, etc. For assistance, therefore, please contact either Marie Paxton or John Litt at AMT Reading.

Table 1 : Set of Standard Keywords

acoustics
ada
algebra
algorithm
alvey
apal
applications
architecture
astrophysics
benchmarks
cad
cartography
character
climate
comparison
data
differential
dna
dynamics
edge
eigen
fe (ie finite elements)
fft
fluid
fractal
graphics
hardware
heat
image
integration
ising
language
lisp
magnetic
matrix
modelling
molecular
monte (ie carlo)
neural
pascal
pdt
performance
prolog
protein
radar
random
sar
seismic
signal
simulation
sorting
speech
statistics
text
vision

Selection is also performed on the words in the Titles of the references.

Ref: JL/DAPREFS

Table 2 : Current List of DAP References (April 1989)

Allen M P 1988 Molecular Graphics and the Computer Simulation of Liquid Crystals. Bristol University, UK7.16-

Allen M P, O'Shea S F 1987 A Monte Carlo Simulation Study of Orientational Domain Clusters in the Planar Quadruple Model. Molecular Simulation 1:47-66

Anthony S J, Bennett M D, Rapley G, Stubbington B D, Thomas M, Thomas W G 1988 A Comparison of Approaches to Parallelism. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Appleby D G, Soraghan J J 1986 Fast SAR Signal Processing on the DAP. Southampton University, UK7.9-

Arnot N R, Wilkinson G G, Burge R E 1982 Applications of the ICL DAP for Two-dimensional Image Processing. Computer Physics Communications 26:455-457

Askew S L, Walkden F 1984 On Programming Parallel Computers to Solve Engineering Fluid Dynamics Problems. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Askew S L, Walkden F 1984 On the Design and Implementation of a Package for Solving a Class of Partial Differential Equations on the ICL DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Babenhauserheide M, Pecht J 1982 PPSTAR - A Fortran IV Software Package to Support the Development of Portable Image Processing Software. IEEE 6th Annual Conference on Pattern Recognition, , Ed., , Munich-

Bale A 1986 Implementing LISP on the ICL DAP. QMC MSc Thesis, UK14.1-

Barlow R H, Evans D J, Shanehchi J 1984 Sparse Matrix Vector Multiplication on the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Bates P A, Lawes R A 1983 RIF, a Generalised Software System for Vector Scan Machines. in: Microcircuit Engineering, , Ed., Cambridge, -

Bond A H 1984 The Use of Programmable Broadcast Array Architecture for Computer Vision. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Bowgen G S J 1987 Delayed Execution of DAP Programs. , AMT TR2-

Bowgen G S J, Modi J J 1985 Implementation of QR Factorization on the DAP Using Householder Transformations. Computer Physics Communications 37:167-170

Bowler K C, Bruce A D, Kenway R D, Pawley G S, Wallace D J 1987 Exploiting Highly Concurrent Computers for Physics. Physics Today 1987:40-48

Bowler K C, Bruce A D, Kenway R D, Pawley G S, Wallace D J, McKendrick A 1987 Scientific Computation on the Edinburgh DAPs. Edinburgh University (Dec 1987), UK13.15-

Bowler K C, Pawley G S 1984 Molecular Dynamics and Monte Carlo Simulations in Solid State and Elementary Particle Physics. Proceedings IEEE 72:42-55

Bownes M, Shirras A, Blair M, Collins J, Coulson A 1988 Evidence that Insect Embryogenesis Is Regulated by Ecdysteroids Released from Yolk Proteins. Proc. National Academy of Science USA 85:1554-1557

Bruce A D, Canning A, Forrest B, Gardner E, Wallace D J 1986 Learning and Memory Properties in Fully Connected Networks. Edinburgh University 86/366, UK13.12-

Buxton B F, Murray D W, Buxton H, Williams N S 1985 Structure-from-motion Algorithms for Computer Vision on an SIMD Architecture. Computer Physics Communications 37:273-280

Canning A, Gardner E 1988 Partially Connected Models of Neural Networks. Edinburgh University 88/433, UK13.9-

Carroll D M, Pogue C A, Willett P 1986 Bibliographic Pattern Matching Using the ICL Distributed Array Processor. Sheffield University, UK12.3-

Carver G 1988 A Spectral Meteorological Model on the ICL DAP. Parallel Computing 8:121-126

Clarke L J 1988 Magnetic Lattice Simulations on the AMT DAP. Edinburgh (Aug 1988), Alvey ARCH001 Report-

Clint M, Holt C, Perrott R H, Stewart A 1984 Algorithms for the Parallel Computation of Eigensystems. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Clint M, Holt C, Perrott R H, Stewart A 1985 A DAP FORTRAN Subroutine for the Eigensolutions of Real Symmetric Matrices. Computer Journal 28:340-342

Collier W D, McCallien C W J, Enderby J A 1984 Tough Problems in Reactor Design. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Collins J F, Coulson A F W 1984 Applications of Parallel Processing Algorithms for DNA Sequence Analysis. Nucleic Acids Research 12:181-192

Collins J F, Coulson A F W, Lyall A 1988 The Significance of Protein Sequence Similarities. CABIOS 4:67-71

Cooper B 1989 Statistical Tabulation. , AMT AN9-

Cooper B E 1988 The Distributed Array Processor in Statistical Computing. AMT Consultant, UK12.7-

Coulson A F W, Collins J F 1988 High Performance Computation for Data Handling in Molecular Biology. Edinburgh University, UK7.4-

Coulson A F W, Collins J F, Lyall A 1987 Protein and Nucleic Acid Sequence Database Searching: a Suitable Case for Parallel Processing. The Computer Journal 30:420-424

Cowan D 1988 The ESPRIT Application for a Project to Develop a 3-D Image Analysis System for a Laser Scanning Microscope. Edinburgh (Aug 1988), Alvey ARCH001 Report-

Cowan D 1988 Use of the DAP for NMR Image Processing. Edinburgh (Aug 1988), Alvey ARCH001 Report-

Cryer C W, Flanders P M, Hunt D J, Reddaway S F, Stansbury J 1982 The Solution of Linear Complementarity Problems on an Array Processor. Journal of Computational Physics 47:258-280

Davies A J 1988 The Boundary Element Method on the ICL DAP. Parallel Computing 8:335-343

Davies A J 1988 Mapping the Boundary Element Method to the ICL DAP. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Davies S T 1984 The Implementation of the FFT on the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Delves L M 1985 Monte Carlo Calculations of Neutron Diffusion on the ICL DAP. Computer Physics Communications 37:295-301

Delves L M, McCrann M 1987 DAP-Ada: Ada Facilities for SIMD Architectures. ICL Technical Journal 5:778-788

Delves L M, Samba A 1982 Deconvolution of Seismic Data. Computer Physics Communications 26:473-476

Delves L M, Samba A S, Hendry J A 1984 Band Matrices on the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Dixon L C W, Ducksbury P G 1985 Finite Element Optimisation on the DAP. Computer Physics Communications 37:187-193

Dongarra J J 1985 Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment. Argonne National Lab. Tech. Memo 23, UK4.7-

Duller A W G, Paddon D J 1984 Processor Arrays and the Finite Element Method. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Erhard W, Gutzmann M 1988 A Classification of Electroencephalograms on the DAP. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Evans D J 1984 New Parallel Algorithms for Partial Differential Equations. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Evans D J, Shanehchi J, Barlow R H 1984 Implementation of the Conjugate Gradient and Lanczos Algorithms for Large Sparse Banded Matrices on the ICL-DAP. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Fincham D 1984 Molecular Dynamics and Graphics Using the DAP. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Fincham D 1987 Adaptation of the Program MC of Professor K Lucas to the DAP. , UK8.5-

Fincham D 1987 Parallel Computers and Molecular Simulation. Molecular Simulation 1:1-45

Flanders P M 1979 Fortran Extensions for a Highly Parallel Processor. Infotech State of the Art Report on Supercomputers 2:-

Flanders P M 1982 Non-numerical Methods on Parallel Computers. Computer Physics Communications 26:363-371

Flanders P M 1982 A Unified Approach to a Class of Data Movements on an Array Processor. IEEE Transactions on Computers C-31:809-819

Flanders P M 1983 Parallel Algorithms for Basic Data Compression and Expansion on the ICL DAP. , ICL CM85-

Flanders P M 1984 S-APAL. , ICL CM94-

Flanders P M 1985 A Methodology for Emulating Sorting and Routing Networks on Processor Arrays. , ICL CM97-

Flanders P M 1985 A General Purpose Sort Program in S-APAL. , ICL CM96-

Flanders P M 1985 On the Design of the Machine Code Level for a Range of DAPs. , ICL CM95-

Flanders P M 1986 Towards Array-size Independence on DAP. , ICL CM101-

Flanders P M 1986 S-TRACE: Structured Trace Facilities in S-APAL. , ICL CM102-

Flanders P M 1986 Array Processing - Languages and Implementation. Languages for Parallel Processing, , Ed., BCS Parallel Processing Group, Imperial College London-

Flanders P M 1987 Image Magnification. AMT-PMF/007, UK7.11-

Flanders P M 1987 Notes on Automatic Generation of PDTs. AMT-PMF/020, UK8.16-

Flanders P M 1987 Using the Parallel Data Transform Software. AMT-PMF/002, UK8.17-

Flanders P M 1987 Development of S-APAL and PDT Software. AMT-PMF/011, UK8.18-

Flanders P M 1988 The Effective Use of SIMD Processor Arrays. The Design and Application of Parallel Digital Processors, , Ed., IEE, Lisbon Portugal-

Flanders P M 1988 An Overview of Virtual Systems Architecture. , AMT TR15-

Flanders P M 1988 Sorting on Processor Arrays. , AMT TR17-

Flanders P M, Hunt D J, Parkinson D, Reddaway S F 1977 Experience Gained in Programming the Pilot DAP,a Parallel Processor with 1024 Processing Elements. in: Parallel Computers - Parallel Mathematics, , Ed., North Holland, -

Flanders P M, Hunt D J, Reddaway S F, Parkinson D 1977 Efficient High Speed Computing with The Distributed Array Processor. in: High Speed Computer and Algorithm Organization, Kuck D J, Lawrie D H, Sameh A H, Eds., , Academic Press-

Flanders P M, Parkinson D 1987 Data Mapping and Routing for Highly Parallel Processor Arrays. Future Computing Systems 2:183-224

Flanders P M, Parkinson D 1987 Data Organization for the Effective Use of Processor Arrays. Parallel Processing by Cellular Automata and Arrays, , Ed., 3rd International Workshop, Berlin Sept 1986-

Flanders P M, Reddaway S F 1984 Sorting on DAP. ICL Technical Journal 4:139-145

Flanders P M, Reddaway S F 1984 Sorting on DAP. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Fleming I 1981 Background for the Study into the Feasibility of Applying DAP Technology to Multi-mode Airborne Radar. Speech Notes, UK5.10-

Forrest B M, Roweth D, Stroud N, Wallace D J, Wilson G V 1987 Implementing Neural Network Models on Parallel Computers. Edinburgh University 87/414, UK12.2-

Forrest B M, Roweth D, Stroud N, Wallace D J, Wilson G V 1988 Neural Network Models. Parallel Computing 8:71-83

Fox G C 1988 What Have We Learnt from Using Real Parallel Machines to Solve Real Problems. 3rd Conference on Hypercube Concurrent Computers and Applications (Pasadena), UK12.11-

Fragnelli V 1984 Architectures for Sparse Band Matrix Dense Vector Multiplication. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., North Holland, Elsevier Science Publishers-

Gajjar J S B 1985 DAP as a Tool for Numerical Flow Modelling, Part I - Two Dimensional Flow Problems. NMI Ltd Report R195 (Jan 1985), UK14.2-

Gajjar J S B 1985 DAP as a Tool for Numerical Flow Modelling, Part II - Three Dimensional Flows. , UK14.3-

Gajjar J S B 1985 On Some Solutions of the Navier-Stokes Equations Using a Parallel Processor. Computer Physics Communications 37:303-309

Galli L, Resta G 1984 Implementation of a Data Flow Algorithm for Linear Programming. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Gallopoulos E J 1985 The Massively Parallel Processor for Problems in Fluid Dynamics. Computer Physics Communications 37:311-315

Gardiner H, Lyttle R W, Milligan P, Perrott R H 1987 Quick Language Implementation. ICL Technical Journal 5:789-806

Gent C R, Hellier R L, Reddaway S F, Thomas B G 1988 Applying Parallel Processing to an ESM Suite. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Genz A C 1982 Numerical Multiple Integration on Parallel Computers. Computer Physics Communications 26:349-352

Genz A, Swayne D A 1984 Parallel Implementation of Alod Methods for Partial Differential Equations. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Ginsburg M 1988 Database Searching: a Short Comparative Study. Imperial Cancer Fund Report, UK12.10-

Godfrey M D 1986 Innovation in Computational Architecture and Design. ICL Technical Journal 5:18-31

Gostick R W 1979 Software and Algorithms for the Distributed-Array Processors. ICL Technical Journal 1:116-135

Gostick R W 1981 Software and Hardware Technology for the ICL DAP. Australian Computer Journal 13:1-6

Grosch C E 1986 Adapting a Navier-Stokes Code to the ICL-DAP. , UK2.5-

Harding A F, Carling J C 1984 The Three-dimensional Solution of the Equations of Flow and Heat Transfer in Glass-melting Tank Furnaces: Adapting to the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Healey A R, Davies S T 1984 Statistical Model Fitting on the ICL Distributed Array Processor. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Hellier R L 1982 DAP Implementation of the WZ Algorithm. Computer Physics Communications 26:321-323

Hellier R L 1988 Ray Tracing. , AMT AN6-

Hendry J A, Delves L M 1984 GEM Calculations on the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Hockney R W 1982 Characterization of Parallel Computers and Algorithms. Computer Physics Communications 26:285-291

Hockney R W 1984 Optimizing the FACR(l) Poisson-solver on Parallel Computers. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Holt C M, Stewart A, Clint M, Perrott R H 1987 An Improved Parallel Thinning Algorithm. Communications of the ACM 30:156-160

Horn A 1986 Demonstrating Tracking on the DAP. , UK8.1-

House S R 1984 Symbol Processing on the Distributed Array Processor. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Howlett J, Parkinson D, Sylwestrowicz J 1983 DAP in Action. ICL Technical Journal 3:330-344

Hubbard R E, Fincham D 1985 Shaded Molecular Surface Graphics on a Highly Parallel Computer. Journal of Molecular Graphics 3:12-15

Hubbard R, Fincham D 1983 Molecular Graphics Using the DAP. CCP5 Newsletter 11:-
Hunt D J 1981 Solution of a Large System of Equations on DAP Using a Hybrid Gauss/Gauss-Jordan Method. , ICL CM75-
Hunt D J 1981 The ICL DAP and Its Application to Image Processing. in: Languages and Architectures for Image Processing, Duff, Levialdi, Eds., , Academic Press-
Hunt D J 1981 Implementation of Choleski Decomposition on DAP. , ICL CM76-
Hunt D J 1983 Tracking of LSI Chips and Printed Circuit Boards Using the ICL DAP. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-
Hunt D J 1983 Decoding of Quadtree Image Representations on DAP. , ICL CM86-
Hunt D J 1983 Test Generation Methods and the Applicability of DAP. , ICL CM92-
Hunt D J 1983 Performance Comparisons of Some DAP Options with Hierarchical Store. , ICL CM91-
Hunt D J 1983 Input-Output Character Conversions on DAP. , ICL CM88-
Hunt D J 1983 Smooth Line Drawing by DAP. , ICL CM87-
Hunt D J 1983 Outline of the DAP Tracking Programs. , ICL CM83-
Hunt D J 1983 DAP Sorting Applied to Trade Statistics Data. , ICL CM84-
Hunt D J 1984 Hardware for Fast Input-output Data Re-organisation. , ICL CM81-
Hunt D J 1987 Gate Array Routing on the AMT DAP 500 Processor Array. CAVE Workshop Paper (Spain), UK8.2-
Hunt D J 1987 'Bigsolve' Timings on MiniDAP. , AMT TR5-
Hunt D J 1988 Gate Array Routing. , AMT AN3-
Hunt D J 1989 AMT DAP - a Processor Array in a Workstation Environment. Computer Systems Science and Engineering 4:107-114
Hunt D J 1989 Implicit Solutions for Oversize Problems. , AMT TR24-
Hunt D J, Reddaway S F 1983 Distributed Processing Power in Memory. in: The Fifth Generation Computer Project, , Ed., Maidenhead, Pergamon Infotech-
Hunt D J, Webb S J, Wilson A 1980 Application of a Parallel Processor to the Solution of Finite Difference Problems. , ICL CM71-
James R A 1982 Simulation of Particle Problems in Astrophysics. Computer Physics Communications 26:423-431
James R A, Parkinson D 1980 Simulation of Galactic Evolution on the ICL Distributed Array Processor. IUCC Bulletin 2:111-114
Jesshope C R 1980 Data Routing and Transpositions in Processor Arrays. ICL Technical Journal 2:191-206
Johns T C, Nelson A H 1985 Particle Simulation of 3D Galactic Hydrodynamics on the ICL DAP. Computer Physics Communications 37:329-336

Jong C C, Pearmain A J, Coward P R 1987 DAPMAC: Design-rule Checking Using a Distributed Array Processor. VLSI and Computers COMP EURO 87, , Ed., , Hamburg May 1987-

Kacsuk P 1988 DAP Prolog: a Parallel Array Extension of Prolog. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Kacsuk P, Bale A 1987 DAP Prolog. VAPP III, , Ed., , Liverpool-

Kashko A 1987 A Parallel Approach to Graduated Nonconvexity on an SIMD Machine. , UK9.3-

Kashko A, Buxton H, Buxton B F, Castelow D A 1988 Parallel Matching and Reconstruction Algorithms in Computer Vision. Parallel Computing 8:3-17

Kirby P 1981 A Moving Mesh Plasma Equilibrium Problem on the ICL DAP. ICL Technical Journal 2:403-424

Kosko B 1987 Constructing an Associative Memory. Byte 1987:137-144

Krzeczkowski A J, Smith E A, Gethin T 1982 Seismic Migration Using the ICL Distributed Array Processor. Computer Physics Communications 26:447-453

Kuehn J T 1984 Parallel C Language Extensions for PASM. Purdue University, UK4.14-

Lai C H, Liddell H M 1987 Preconditioned Conjugate Gradient Methods on the DAP. Queen Mary College, UK8.6-

Lai C H, Liddell H M 1987 A Review of Parallel Finite Element Methods on the DAP. Queen Mary College, UK7.18-

Lai C H, Liddell H M 1988 Finite Elements Using Long Vectors of the DAP. Parallel Computing 8:351-361

Lawson A C 1988 Adapting a Serial Acoustics Code to Run on the AMT DAP. , AMT TR12-

Lawson A C 1988 Assessment of the Impact of Using Different Data Lengths When Running on Engineering Application on the AMT DAP 510. , AMT TR13-

Liddell H M, Bowgen G S J 1982 The DAP Subroutine Library. Computer Physics Communications 26:311-315

Liddell H M, Parkinson D 1989 Mapping Large Scale Computational Problems on a Highly Parallel SIMD Computer. Dec 1987 SIAM Conference on Parallel Processing for Scientific Computing, Rodrigue G, Ed., SIAM Philadelphia-

Liddell H M, Parkinson D, Wait R 1987 A Parallel Computational Environment for Finite Element Calculations. , UK7.17-

Ljuslin C, Lone S 1988 Evaluation of a SIMD Architecture in HEP-triggering. CERN-LAA RT/88-03, UK12.5-

Longstaff I D 1984 The Military DAP Programme in the UK. Advanced Signal Processing Seminar (Warwick), UK4.12-

Lyall A, Hill C, Collins J F, Coulson A F W 1986 Implementation of Inexact String Matching Algorithms on the ICL DAP. in: Parallel Computing 85, Feilmeier M, Joubert G, Schendel U, Eds., , Elsevier Science Publ-

Lynch M F, Willett P 1987 Information Retrieval Research in the Department of Information Studies, Sheffield University 1965-1985. Journal of Information Science 13:221-234

Lynch M, Willett P 1987 Current Research into Chemical and Textual Information Retrieval at the Department of Information Studies, Sheffield University. Information Processing and Management 23:447-463

MacQueen K S 1988 Performance Estimates for 4K FFTs. , AMT TR20-

MacQueen K S 1988 Fast 1K Gather, Scatter and Sort Routines. , AMT TR21-

MacQueen K S 1988 High Performance Batched FFT Examples. , AMT TR22-

Manning L J, Dew P M, Wang H 1987 Programming Models for VLSI Array Processors with Application to Low-Level Vision Processing (Draft). Parallel Architecture and Computer Vision Workshop Oxford 1987, UK11.15-

Martin B R, Dunford E 1982 Processing Astronomical Images from Space. Computer Physics Communications 26:441-446

McCarey I 1988 Cleaning Up: How Parallel Processing Will Help Control Pollution in the North Sea. Parallelogram 7:1-5

McKerrell A, Delves L M 1984 Solution of the Global Element Equations on ICL DAP. ICL Technical Journal 4:50-58

McKerrell A, Delves L M 1988 Monte Carlo Simulation of Neutron Diffusion on SIMD Architectures. Parallel Computing 8:363-370

Merrifield B C, Roberts J B G, Simpson P, Stanley A 1988 Real Time Applications of DAP. 3rd International Conference on Supercomputing, , Ed., ISI Inc, Boston-

Modi J J, Bowgen G S J 1984 QU Factorisation and Singular Value Decomposition on the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Modi J J, Davies R O, Parkinson D 1984 Extension of the Parallel Jacobi Method to the Generalised Eigenvalue Problem. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Modi J J, Parkinson D 1982 Study of Jacobi Methods for Eigenvalues and Singular Value Decomposition on DAP. Computer Physics Communications 26:317-320

Morjaria M, Makinson G J 1984 Unstructured Sparse Matrix Multiplication on the DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Morley R E, Christensen G E, Sullivan T J, Kamin O 1988 The Design of a Bit-serial Coprocessor to Perform Multiplication and Division on a Massively Parallel Architecture. Washington University Report, UK12.12-

Mouhas C N 1987 Automatic Mesh Generation on the DAP. VAPP III, , Ed., , Liverpool-

Munzel G 1984 Residue Arithmetic for Exact Calculations on the DAP. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Oldfield D E 1984 Document Abstracting on the Distributed Array Processor. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Oldfield D E, Reddaway S F 1985 An Image Understanding Performance Study on the ICL Distributed Array Processor. Workshop on Computer Architecture for Pattern Analysis and Image Data Base Management, , Ed., IEEE, Florida-

Page R M R, Baddiley E 1986 Suggested Extension of ICL DAP Parallelism. ICL Technical Journal 5:158-162

Page R M R, Reddaway S F 1987 The DAP as a Filestore Search Engine. AMT TR3, -

Parkinson D 1982 Practical Parallel Processors and Their Uses. in: Parallel Processing Systems:An Advanced Course, Evans D J, Ed., , Cambridge University Press-

Parkinson D 1982 Using the ICL DAP. Computer Physics Communications 26:227-232

Parkinson D 1983 The Distributed Array Processor (DAP). Computer Physics Communications 28:325-336

Parkinson D 1984 The Solution of N Linear Equations Using P Processors. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Parkinson D 1987 Organisational Aspects of Using Parallel Computers. Parallel Computing 5:75-83

Parkinson D 1988 Reducing Edge-size Dependence. , AMT TR8-

Parkinson D 1988 Performance Estimation in a Highly Parallel Environment. UNICOM Seminar on Evaluating Supercomputers, , Ed., , London (June 1988)-

Parkinson D 1989 A Fully Worked DAP Example Program. , AMT TR23-

Parkinson D 1989 Super Parallel Algorithms. , AMT TR25-

Parkinson D, Hunt D J, MacQueen K S 1988 The AMT DAP 500. 33rd IEEE Computer Society International Conference, , Ed., IEEE, San Francisco196-199

Parkinson D, Liddell H M 1983 The Measurement of Performance on a Highly Parallel System. IEEE Transactions on Computers C-32:32-37

Parkinson D, Thanakij R 1988 An Exercise in Optimization in Fortran-Plus. , AMT TR14-

Parkinson D, Wunderlich M 1984 A Compact Algorithm for Gaussian Elimination over GF(2) Implemented on Highly Parallel Computers. Parallel Computing 1:65-73

Patel K D 1984 Implementation of a Parallel (SIMD) Modified Newton Algorithm on the ICL DAP. in: Supercomputers and Parallel Computation, Paddon D J, Ed., Bristol Workshop 1982, Oxford University Press-

Pavelin C J 1988 DAP General Description. , AMT AN1-

Pawley G S 1988 Scientific Applications of the DAP. 3rd International Conference on Supercomputing, , Ed., ISI Inc, Boston-

Pawley G S 1988 Polypeptide Conformation Modelling. Edinburgh (Aug 1988), Alvey ARCH001 Report-

Pawley G S 1988 Notes on the Connection Machine. Edinburgh(July 1988), Alvey ARCH001 Report-

Pawley G S, Bowler K C, Kenway R D, Wallace D J 1985 Concurrency and Parallelism in MC and MD Simulations in Physics. Computer Physics Communications 37:251-260

Pawley G S, Thomas G W 1982 The Implementation of Lattice Calculations on the DAP. Journal of Computational Physics 47:165-178

Pearmain A J, Jong C C, Coward P R 1987 Using the DAP in a CAD System. EDA Conference Wembley, UK8.4-

Pecht J, Ramm I 1982 Recognition of Hand-written Characters Using the DAP. ICL Technical Journal 3:199-217

Perrott R H 1988 Notes on the Language Features of FORTRAN 8X and Their Relevance to DAP FORTRAN. , UK9.6-

Perrott R H, Lyttle R W, Dhillon P S 1987 The Design and Implementation of a Pascal-based Language for Array Processor Architectures. Journal of Parallel and Distributed Computing 4:266-287

Perrott R H, Zarea-Aliabadi A 1986 Supercomputer Languages. Computing Surveys 18:5-22

Pogue C A, Rasmussen E M, Willett P 1988 Searching and Clustering of Databases Using the ICL Distributed Array Processor. Parallel Computing 8:399-407

Pogue C A, Willett P 1984 An Evaluation of Document Retrieval from Serial Files Using the ICL DAP. Online Review 8:569-584

Pogue C A, Willett P 1987 Text Searching Algorithms for Parallel Processors. British Library, UK6.16-

Pogue C A, Willett P 1987 Use of Text Signatures for Document Retrieval in a Highly Parallel Environment. Parallel Computing 4:259-268

Pryde G C, Delves L M, Luttrell S P 1988 A Comparative Study of the AMT DAP and of Transputer Array Architectures for the Super-resolution of Synthetic Aperture Radar Images. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Quinn J E 1988 Molecular Graphics. , AMT AN4-

Quinn J E 1989 Electronic Computer Aided Design. , AMT AN7-

Rasmussen E M, Willett P 1987 Non-hierarchic Document Clustering Using the ICL Distributed Array Processor. 10th Annual International Conference on Research and Development in Information Retrieval, Yu C T, Van Rijsbergen C J, Eds., BCS and AICA, New Orleans Louisiana-

Reddaway S F 1978 DAP - a Flexible Number-cruncher. LASL Workshop on Vector and Parallel Processors, , Ed., , Los Alamos-

Reddaway S F 1979 The DAP Approach. , UK3.11-

Reddaway S F 1980 Revolutionary Array Processors. in: Electronics to Microelectronics, Kaiser W A, Proebster W E, Eds., , North Holland-

Reddaway S F 1983 DAP and Its Application to Image Processing Tasks. Proc Image Processing Symposium:RSRE Malvern-

Reddaway S F 1984 Distributed Array Processor, Architecture and Performance. in: NATO ASI Series Vol F7, Kowalik J S, Ed., Berlin, Springer-Verlag-

Reddaway S F 1985 Median Filtering on DAP. , CM98-

Reddaway S F 1987 Signal Processing on a Processor Array. in: Signal Processing, Lacoume J L, Durrani T S, Stara R, Eds., , Elsevier Science Publishers883-857

Reddaway S F 1988 Mapping Images onto Processor Array Hardware. in: Parallel Architecture and Computer Vision, Page I, Ed., , OUP-

Reddaway S F 1988 A Fast 'Put-dots'. , AMT TR9-

Reddaway S F 1988 Fast Interactive Ising Codes on the AMT DAP510. overhead transparencies of a paper to the SIAM Annual Meeting (July 1988), UK14.6-

Reddaway S F 1988 Fast Evaluation of the Fractal Dimension of Boolean Images. overhead transparencies of a paper to the SIAM Annual Meeting (July 1988), UK14.7-

Reddaway S F 1988 Achieving High Performance Applications on the DAP. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Reddaway S F, Bowgen G, van den Berghe S 1989 High Performance Linear Algebra on the AMT DAP 510. Dec 1987 SIAM Conference on Parallel Processing for Scientific Computing, Rodrigue G, Ed., SIAM Philadelphia-

Reddaway S F, Flanders P M 1982 Sorting on DAP. , ICL CM72-

Reddaway S F, Hunt D J, Flanders P M 1979 Application of the ICL DAP. Convention Informatique, , Ed., , Paris-

Reddaway S F, Page R M R 1988 High Speed Data Searching with a Processor Array. Microprocessing and Microprogramming 24:655-660

Reddaway S F, Roberts J B G, Simpson P, Merrifield B C 1985 Distributed Array Processors for Military Applications . Proc MILCOMP 85:,,-Microwave Exhibition and Publishers Ltd

Reddaway S F, Scott D M, Smith K A 1985 A Very High Speed Monte Carlo Simulation on DAP. Computer Physics Communications 37:351-356

Reddaway S F, Wilson A 1988 Regeneration of Images from IFS Codes on a Processor Array. poster paper to SIAM Annual Meeting, UK13.16-

Reddaway S F, Wilson A, Horn A 1989 Fractal Graphics and Image Compression on a SIMD Processor. Frontiers '88 Conference October 1988, , Ed., IEEE Computer Society-

Roberts J B G 1983 The Impact of VLSI on the Architectures of Signal Processing Systems. RSRE Malvern, UK5.3-

Roberts J B G 1986 Highly Parallel SIMD and MIMD Programmable Processors for Signal Processing and Simulation in Defence. , GM3-

Roberts J B G 1987 Digital Signal Processing in the UK. Electronics and Power 1987:183-186

Roberts J B G, Harp J G, Merrifield B C 1986 Real Time Processing with SIMD and MIMD Multiprocessors. RSRE, UK7.10-

Roberts J B G, Harp J G, Merrifield B C, Palmer K J, Simpson P, Ward J S, Webber H C 1988 Evaluating Parallel Processors for Real Time Applications. Parallel Computing 8:245-254

Roberts J B G, Simpson P, Merrifield B C 1985 Applying Digital VLSI Technology to Radar Signal Processing. RSRE Malvern, UK4.8-

Roberts J B G, Simpson P, Merrifield B C, Cross J F 1984 Signal Processing Applications of a Distributed Array Processor. IEE Proceedings 131:603-609

Ruff B P D 1987 A Pipelined Architecture for a Video Rate Canny Operator Used at the Initial Stage of a Stereo Image Analysis System. , UK11.16-

Ruffhead A 1985 Parallel Processing Applied to Digital Terrain Matrices. Computer Physics Communications 37:357-361

Schmid H J 1984 A DAP-Fortran Code for the Hungarian Method. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Sharman K C 1988 Neural Networks on the DAP. Strathclyde University, UK12.4-

Simpson P, Merrifield B C 1985 Real Time Signal Processing Applications of a Distributed Array Processor,. Computer Physics Communications 37:133-140

Simpson P, Merrifield B C, Roberts J B G 1988 Terrain Map Calculations on DAP. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Simpson P, Roberts J B G 1983 Real Time Speech Recognition on a Distributed Digital Processing Array. RSRE Malvern Memo 3643, UK5.1-

Simpson P, Roberts J B G 1983 Speech Recognition on a Distributed Array Processor. Electronic Letters 19:1018-1020

Simpson P, Roberts J B G, Merrifield B C 1985 Mil-DAP: Its Architecture and Role as a Real Time Airborne Digital Signal Processor. NATO AGARD Conference Pub 380, , Ed., , Lisbon-

Smith K A 1984 An Image Manipulation Package for the DAP. in: Parallel Computing 83, Feilmeier M, Joubert G, Schendel U, Eds., Amsterdam, Elsevier Science Publishers-

Smith K A, Reddaway S F, Scott D M 1985 Very High Performance Pseudo-random Number Generation on DAP. Computer Physics Communications 37:239-244

Soraghan J J, Appleby D G 1986 Use of a Processor Array for SAR Processing. Southampton University, UK7.8-

Strey A 1988 Combinatorial Optimization on an SIMD Parallel Computer. CONPAR'88, , Ed., BCS, Manchester Sept 1988-

Swanson J A, Cameron G R, Haberland J C 1983 Adapting the Ansys Finite-Element Analysis Program to an Attached Processor. IEEE Computer :85-91

Sylwestrowicz J D 1981 Applications of ICL DAP in Econometric Computations. ICL Technical Journal 2:280-286

Thompson B R 1988 Solving the Navier-Stokes Equations. , AMT AN2-

Thompson B, Reddaway S F 1988 Video Conferencing Using Parallel Processing. ERA Technology Report 88-0386 on Digital Signal Processing: Components and Applications, UK13.14-

Tinsley C J 1987 Parallel Monte Carlo Studies of Ising and Classical Heisenberg Models of a Spinel Ferromagnet. VAPP III, , Ed., , Liverpool-

Tinsley C J 1987 A Monte Carlo Investigation of the Dependence of Spontaneous Magnetization on Temperature in Ising Models of Lithium Ferrite. Philosophical Magazine B 56:351-360

Toral R, Wall C 1987 Finite Size Scaling Study of the Equilibrium Cluster Distribution of the Two-dimensional Ising Model. Journal of Physics A20:4949-

van den Berghe C S 1988 The Application of SIMD Processing to Problems in Vehicle Dynamics. , AMT TR11-

van den Berghe C S 1988 Linear Algebra: LL' Decomposition. , AMT AN5-

van den Berghe C S 1988 Meteorological Modelling on the DAP Series of Computers. ECMWF Workshop on Use of Parallel Processors in Meteorology, UK12.9-

van den Berghe C S 1989 Molecular Dynamics. , AMT AN8-

van den Berghe S 1987 Fortran Plus Timings. , AMT TR1-

Wait R 1988 Partitioning and Preconditioning of Finite Element Matrices on the DAP. Parallel Computing 8:275-284

Wait R, Landauro J 1987 Parallel Algorithms for Multicomponent Separation Calculations. VAPP III, , Ed., , Liverpool-

Wallace D J 1985 Spin Glass Models of Neural Networks: Size Dependence of Memory Properties. Edinburgh University 85/352, UK13.13-

Webb S J 1980 Solution of Elliptic Partial Differential Equations on the ICL Distributed Array Processor. ICL Technical Journal 2:175-190

Webb S J, McKeown J J, Hunt D J 1982 The Solution of Linear Equations on a SIMD Computer Using a Parallel Iterative Algorithm. Computer Physics Communications 26:325-329

Weston J S 1985 Two Algorithms for the Parallel Computation of Eigenvalues and Eigenvectors of Large Symmetric Matrices using the ICL DAP. , UK9.4-

Wilkinson G G, Burge R E 1982 Using the ICL DAP for Satellite Climatology. Computer Physics Communications 26:469-471

Williams N S, Buxton B F, Buxton H 1987 Distributed Ray Tracing Using an SIMD Processor Array. BP Research Centre, -

Wilson A 1981 Array Processing - Principles and Practice. ACM81 paper, UK5.8-

Wilson A 1986 Experience with programming parallel signal processing algorithms in Fortran 8X. ICL Technical Journal 5:344-350

Wilson G 1988 Computing in Parallel. in: New Scientist Feb 1988, , , -

Winkler K H A, Chalmers J W, Hodson S W, Woodward P R, Zabusky N J 1987 A Numerical Laboratory. Physics Today 1987:28-37

Zacharov V 1982 Parallelism and Array Processing. CERN School of Computing Lecture in Zinal, UK5.4-

Zhao C, Wood J 1988 The Monte Carlo Method on a Parallel Computer. Queen Mary College, UK8.15-

NEWS RELEASE

AMT

ACTIVE MEMORY TECHNOLOGY

US ARMY ENGINEERS' WATERWAYS EXPERIMENT STATION ACCEPTS DELIVERY OF AMT DAP 610

READING, 7th June 1989 - Active Memory Technology Limited, announced today that the US Army Engineers' WES, Vicksburg, Mississippi has accepted delivery of an AMT DAP 610 massively parallel computing system valued at $360,000.

The DAP 610 will be mounted in a helicopter and used to process airborne scanner imagery for minefield detection, according to Dan Cress, Technical Project Manager. The technology will be demonstrated in the third quarter of 1990.

"The DAP was brought to our attention by the people at NASA Goddard who are involved in massively parallel computing", Cress said. "The DAP represents a well developed computer architecture with associated memory, bus and I/O structure, around which we can centralise other processing. Furthermore, the DAP supports our data compression and can be attached to asynchronous processors to further expand capability, " Cress added.

Rodney Hornstein Managing Director of AMT said "This is a further important contract AMT has won in the defence market in the USA. It shows that British Companies can win such business in open competition if they can demonstrate they have advanced technology at an attractive price."

The AMT DAP (Distributed Array of Processors) is a fine grain, massively parallel computer which provides defence contractors with the high-speed data rates and processing speed required for radar and sonar processing, speech recognition, neural network research as well as other signal and image processing applications. The DAP shortens system development time, as the same platform is used in system deployment.

EDITORS NOTE

Founded in 1986, AMT is a leading manufacturer of advanced massively parallel computer systems which are supplied as attached processors to Sun and DEC VAX computers, or as OEM products. The systems are designed to achieve very high performance at very low cost. AMT is an International group with development centres in Reading, Berkshire (UK) and Irvine, California (USA).

- END -

For further information please contact: Isabel Harding, Active Memory Technology Limited, 65 Suttons Park Avenue, Reading, Berkshire, RG6 1AZ. Tel : (0734) 661111. Fax : (0734) 351395

Press contact: Stephen Bird, Stephen Bird Associates. Tel : (0734) 774540/332649. Fax : (0734) 333630.

NEWS RELEASE

AMT

ACTIVE MEMORY TECHNOLOGY

LOCKHEED MAP WORLD ON AMT DAP

READING, 20th February 1990 - Active Memory Technology (AMT) has announced an order from Lockheed Aeronautical Systems Company to supply two DAP 610 (Distributed Array Processor) massively parallel computers. The DAPs will be used in a project to convert world maps currently stored on CD ROM (compact disc read-only memory) to erasable optical discs.

According to Lockheed avionics computer system specialist, Norm Kogen, the AMT DAP provides the fastest, most cost effective solution to the problem of compressing an enormous amount of data into a space one tenth of the size currently required. A task said to be impossible using Lockheed's existing computer equipment.

"The DAP is an ideal machine for data compression and reforming tasks which involve several processes" explains Kogen. "First, the shape of the digital map data is manipulated to produce the type of projection required, then colour clustering and run-length encoding are added to complete the process." he said.

The Lockheed project involves converting CDs supplied at 254 pixels per inch, in 24-bit colour, for use at 96 pixels per inch, in 8-bit colour. Now, using optical discs, map details can then be updated and modified according to any particular requirement, a facility impossible using CD ROMs.

EDITORS NOTE

Founded in 1986, AMT is a leading manufacturer of advanced massively parallel computer systems which are supplied as attached processors to Sun and DEC VAX computers, or as OEM products. The systems are designed to achieve very high performance at very low cost. AMT is an International group with development centres in Reading, Berkshire (UK) and Irvine, California (USA). Presently, the company has installed more than 70 systems in seven countries.

- END -

For further information please contact: Isabel Harding, Active Memory Technology Limited, 65 Suttons Park Avenue, Reading, Berkshire, RG6 1AZ. Tel : (0734) 661111. Fax : (0734) 351395

Press contact: Stephen Bird, Stephen Bird Associates. Tel : (0734) 774540/332649. Fax : (0734) 333630.

NEWS RELEASE

AMT

ACTIVE MEMORY TECHNOLOGY

NEW ACTIVE MEMORY TECHNOLOGY COMPUTERS LEAD INDUSTRY IN PRICE/PERFORMANCE

Reading, April 24th 1990 - -Active Memory Technology (AMT) has announced the DAP (Distributed Array of Processors) CP8 range of massively parallel computer systems, setting a new standard for the industry. Based on a new custom 8 bit, VLSI, co-processor chip (CP8) the AMT machines enhance the performance of the cur rent DAP range by up to 1000 per cent.

AMT's chief executive officer, Dr. Geoff Manning, explained:-

"The new DAP/CP8 family improves the price/performance of the DAP by up to 700 per cent. The new peak performance of the DAP 610C is 560 MFLOPS at 32 bit precision, and 20,000 MIPS for 8 bit integer. This provides floating point price/performance as low as £500 per MFLOP and 8 bit integer performance at £15 per MIP. These are dramatic numbers that place AMT in the forefront of massively parallel systems by providing low-cost super computing performance."

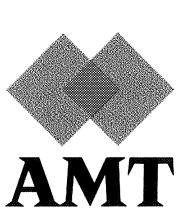
According to Manning, the new DAP/CP8 family will open up new market opportunities, particularly in the areas of simulation and modeling. In signal and image processing, Manning says the DAP/CP8 will be ideal for applications such as seismic processing and synthetic aperture radar processing.

In the DAP/CP8 family of computers systems, an 8 bit co-processor is added to each 1 bit processor in the DAP. The 510C model will contain 1024 8 bit co-processors as well as the 1024 1 bit processors, and the DAP 610C will contain 4096 of both types of processors. . The new 8 bit co-processors are used for complex arithmetic such as floating point operations. The 1 bit processors continue to be used for memory access, data movement, fast input/output and for Boolean or logic operations.

The company says existing users' source code will run unmodified on the new DAP/CP8 family, only a recomplilation is needed to obtain increased performance.

Bill Terry, AMT's international vice president of sales said;

"The new co-processor chips offer an extremely attractive upgrade path for the future for our existing customers. We expect a big demand for the new systems that will be upgraded during the last quarter of this year"



Manning says AMT now has a mature range of products with four computer offerings - the 510, 610, 510C and 610C - spanning a factor of 40 in computer power. The DAP is priced from £87,000 to £300,000 depending on configuration.

AMT offers a wide spectrum of fast input/output devices including high resolution video drivers with 8 or 24 bit colour disks connected at 16 MBytes/sec transfer rate, an input/output computer that can reform data during input or output under program control, and can be connected to remote equipment by optical fibre.

The company say its software is mature and proven and is accepted as very user friendly. A new compiler is available that can deal with arrays of any size, and is even more efficient, the company claims, than its previous offering which was already considered to be excellent.

"In 1986 when AMT started I promised a development path that would provide users with an upwards compatable range that would improve its power by a factor of 1000 in a decade, that is, by a factor of 2 per year" said Manning. " We are well ahead of schedule and we will stay ahead" he added.

EDITORS NOTE:

AMT, a UK funded company, manufactures and markets advanced massively parallel computer systems known as DAPs (Distributed Array of Processors) which are supplied as attached processors to Sun Microsystems and DEC VAX computers, or as OEM products. The DAPs, the third generation of a system which has been in use for over ten years, are designed to achieve very high performance at very low cost. More that 1000 users have worked on a wide range of applications. With over 70 systems in the field, AMT's DAP has a larger installed base than any other massively parallel processor.

NEWS RELEASE

AMT

ACTIVE MEMORY TECHNOLOGY

PARALLEL COMPUTER SYSTEM AIDS CANCER RESEARCH

READING, June 19th 1989

The Imperial Cancer Research Fund has installed a new computer system to give dramatic improvements in the speed and sensitivity of sequence comparison.

The Active Memory Technology Distributed Array Processor (AMT DAP) 610, a massively parallel computer that delivers 40 billion Boolean operations per second, is combined with a new protein database searching program, PROSRCH. It gives ICRF scientists greatly enhanced speed and sensitivity for protein sequence comparison - the task of comparing the sequence of chemical letters of new proteins with the ever expanding protein sequence database.

The new package can cut to around three or four minutes a search which took several hours on ICRF's old system. Its sensitivity can pick up subtle but potentially important comparisons which may previously have been missed - enabling scientists to use the system as a powerful and flexible research tool in their search for how and why things go wrong at subcellular level to cause cancers and other diseases.

The system is initially being used for protein sequence analysis only but researchers at Edinburgh University Biocomputing Research Unit, who produced PROSRCH, are working on a program for DNA sequence comparison, expected to be completed by the end of the year. This will give further possibilities for the much greater task of comparing DNA sequences, the actual genetic information, which may hold the key to understanding many disease processes.

Comparing newly-discovered protein or DNA sequences with known sequences held in computer databases can give the scientists an insight into its possible function and how changes may lead to disease. This insight will prompt further experiments in the laboratory and ultimately new methods of screening, prevention and treatment.

Dr. Ron Catterall, head of ICRF's Research Computing Unit, who initiated the collaboration with AMT and Edinburgh University said:

"This new facility will establish the Imperial Cancer Research Fund as a world leader in sequence analysis with the most powerful system currently in operation. The ability to work in a truly interactive manner, made possible by the speed of the DAP, will generate new modes of research working with important feedback to laboratory studies."

Dr. Catterall and Dr. John Collins of Edinburgh University Biocomputing Research Unit are setting up other collaborative research projects to exploit further the power of the new system in cancer research.

Michelle Ginsburg, a senior scientific officer in ICRF's Mutagenesis Laboratory, who worked with the Edinburgh University group testing the software, said the system was a valuable addition to laboratory work.

"It is a tremendous adjunct to existing tools in trying to find what happens in the cell and why it goes wrong. Sometimes just one change in a protein, for example, can have a devastating effect on its function and hence its role in the cell. If we are able to understand a disease process thoroughly, we can try and do something to correct it. ICRF is taking advantage of up to date technology to help scientists gain new insights in their work."

Rodney Hornstein, Managing Director of AMT said "The DAP is an excellent engine for ultra high speed searching of databases. This is the first of a number of applications in this area which will bring the power of this technology to commercial as well as scientific users."

EDITORS NOTE

Founded in 1986, AMT is a leading manufacturer of advanced massively parallel computer systems which are supplied as attached processors to Sun and DEC VAX computers, or as OEM products. The systems are designed to achieve very high performance at very low cost. AMT is an International group with development centres in Reading, Berkshire (UK) and Irvine, California (USA).

- END -

For further information please contact: Isabel Harding, Active Memory Technology Limited, 65 Suttons Park Avenue, Reading, Berkshire, RG6 1AZ. Tel : (0734) 661111. Fax : (0734) 351395

Press contact: Stephen Bird, Stephen Bird Associates. Tel : (0734) 774540/332649. Fax : (0734) 333630.

Press contact: Sue Kiddy, Press Officer, ICRF. Tel : 01 242 0200.

John Collins, University of Edinburgh. Tel : 031 667 1081 ext 2965.

AMT

Active Memory Technology

Press background information

April 1990

AMT: the facts at a glance

Full name:

Active Memory Technology Limited

European office address:

65 Suttons Park Avenue
Reading
Berkshire RG6 1AZ

USA office address:

Active Memory Technology Inc
16802 Aston Street
Suite 103
Irvine
California 92714

European press contact:

Geoff Manning
Tel: 0734 661111

Press relations:

Stephen Bird
Stephen Bird Associates
0734 332649/774540

Founding date:

31st October 1986

Number of employees:

79 as at 31st March 1990 - 39 in the UK and 40 in the USA.

Product line:

Distributed Array of Processors (DAP)

Massively parallel; known as the DAP family; can be attached to Sun (UNIX-based) and DEC VAX (VMS-based) ranges of equipment

List price:

DAP 510: starts at £87,000
DAP 610: starts at £230,000
DAP 510C: starts at £119,000
DAP 610C: starts at £300,000

Installed base:

DAP 510 range
The DAP 510 machine containing 1024 processors was the first in a range of computers launched by AMT. It represents the third generation of a system which has been in use for over a decade with more than 1000 users having worked on a wide range of applications. AMT shipped its first DAP 510 machine in October 1987 within twelve months of start-up.

DAP 610 range
The first DAP 610 machines containing 4096 processors were shipped in November 1988 one year after the first DAP 510. The performance improvement offered by the DAP 610 marked a four-fold increase in power.

DAP/CP8 range
Eighteen months later the DAP/CP8 range (with 8-bit co-processors) further improves performance of both the DAP 510 and DAP 610 systems by up to 1000% and improves the price-performance by up to 700%.

With over 75 systems in the field, DAP has a larger installed base than any other massively parallel computer.

Critical points of product differentiation:

DAP technology represents a breakthrough in computing price-performance and overcomes many of the limitations of parallel computing.

Current DAP models contain either 1024 or 4096 processors, all of which simultaneously execute the same program but with separate data streams.

Able to accelerate many software applications by factors of 10 to 100, the system comes with a wide range of library routines to ease the development of programs in image processing, signal processing and general scientific and engineering applications.

Speed:
In most applications, DAP speeds up calculations by up to 100 times, allowing problems that used to take hours to be solved in as many minutes.

High speed input/output:
The DAP architecture is uniquely adapted to very fast I/O speeds with minimal overheads on the processing capability. Applications ranging from radar signal processing to large scale free-text search have capitalised on this feature.

Data visualisation:
Due to the DAP's very fast I/O computation results can be displayed in colour almost immediately after they are generated. Other machines with more limited I/O capability are forced to either slow the computation rate to match their I/O capability or produce the picture after the calculations have finished. Already this feature of the DAP has enabled researchers to observe new phenomena by watching the progress of a DAP simulation.

Low cost:
DAP's innovative design enables users to harness 1024 processors for less than £100,000 with performance superior than machines costing at least 10 times more.

Size:
A 1024-processor DAP occupies less than 2 square feet of floor space and can be used in an ordinary office environment as opposed to an air-conditioned computer room.

Ease of use:
DAP can be attached to a standard VAX or Sun host computer and is programmed in FORTRAN, the most widely-used scientific language. Additional languages such as C are planned.

Deployment
Once applications are developed, the DAP can run without the requirement for a host computer. This combined with its physical characteristics make it highly suitable for use in embedded systems.

Distribution:

OEMs and direct sales to technical end-users, universities and research organisations.

Overseas distribution:

Germany: APTEC Computer Systems GmbH, Munich

Currently AMT is negotiating agreements to establish distributors in Japan and Australia.

Sales Representatives

The Netherlands:	Huisman Informatie Systemen (HIS) BV
Switzerland:	Vector Technologies

US sales offices:

AMT Inc
12300 Twinbrook Parkway
Suite 600
Rockville
Maryland 20852

tel: (301) 770 3440

AMT Inc
2624 Briar Patch Lane
Flower Mound
Texas 75028

tel: (817) 430 0211

AMT Inc
16802 Aston St
Suite 103
Irvine
California 92714

tel: (714) 261 8901

AMT Inc
525 Bridgeport Avenue
Suite 105
Shelton
Connecticut 06484

tel: (203) 925 1075

Major OEM and end-users:

E-Systems (USA), US Armed Forces, Lockheed, a major defence contractor, Plessey, Imperial Cancer Research Fund, an international news agency, Argonne National Laboratory (USA), Rutherford Appleton Laboratory (Oxford), Edinburgh University, Queen Mary and Westfield College (University of London).

AMT has achieved a remarkable first in shipping systems in the US for defence applications. This was achieved by trading from day 1 as an international company, so establishing the world lead that AMT currently has in massively parallel computing.

Senior management team:

AMT has recruited an exceptional management team and board of directors in the US and UK including:

Neil Pearce: Non-executive Chairman
Holds non-executive positions in a variety of UK companies

Dr Geoff Manning: Chief Executive
Formerly Director of the Rutherford Appleton Laboratory

Ronald McKellar: Finance Director
Formerly Finance Director of Cirkit Holdings Plc

Bill Terry: VP Sales
Founder of and formerly with Lear Siegler's terminals division

Bruce Alper: VP Corporate Development
Held senior positions with computer companies in the UK and US.

Major investors:

AMT has raised over £10M from some of Europe's leading venture capitalists. 20% of the company is vested with employees and ICL has a 20% holding of AMT in return for the transfer of the intellectual property rights and patents. The major institutional investors are: Advent Capital, Alan Patricof Associates, Baring Brothers Hambrecht & Quist, Electra Investment Trust, Murray Johnstone, 3i, Baring Brothers, ICL, Sharp Technology Fund, Mercury Asset Management, County NatWest Ventures, and Schroder Ventures.

Parallel processing, the AMT approach:

The trend to parallel processing, although still in its infancy, represents a significant shift of emphasis in the computer industry. The approach adopted by AMT is based upon over a decade of research and development; with such unrivalled experience, AMT has established itself as a dominant player in an emerging market reckoned to be worth an estimated potential of £1 billion.

A major key to AMT's increasing success is its strategy to provide low-cost, high powered computing for its customers. The products use proprietary parallel-processor chips and industry-standard hardware, and are designed for integration with popular host workstations or mainframes such as DEC VAX or Sun Microsystems computers.

"AMT is at the forefront of a computing revolution that starts where conventional machines are at their limits," observes AMT CEO, Geoff Manning. "AMT has a mature range of products spanning a factor of 40 in compute power. DAP massively parallel systems are following a development path that will increase their power by 1000 times during the 1990s. For end users our aim is to place low-cost, high powered, very fast computing systems at the fingertips of users in the friendly environment of their personal workstation. To achieve this we are committed to doubling performance every year and to keeping AMT at the forefront of parallel processing. For OEM users AMT offers a single compatible hardware and software computing environment from algorithm development, through prototyping, to a final embedded solution."

Market applications:

Although the DAP is flexible enough to process effectively almost any problem involving large volumes of data, AMT is focusing its marketing effort in three main application areas:

Signal and image processing:
Signals from sources such as radar, satellite pictures or advanced medical scanners must be processed at high speed in many military and civil applications. DAP is an effective alternative to expensive custom-designed electronic systems and greatly reduces development times and design risks.

Simulation and modelling:
DAP is ideal for electronic logic simulation, modelling of advanced materials, complex structural design and biomedical applications such as DNA sequencing.

Text and databases:
DAP is capable of handling text at an extraordinary rate - over 1 billion characters per second. As the need for fast sorting and searching of large databases grows, DAP is an ideal 'supercharger' for such systems.

VAX and VMS are trademarks of Digital Equipment Corporation
UNIX is a trademark of A T & T Bell Laboratories
Sun is a trademark of Sun Microsystems Inc