EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/EF/MPPC 89-1
28 July 1989

PROPOSAL - THE MPPC PROJECT

Massively Parallel Processing Collaboration
ASPEX Microsystems–Brunel University–CEA-CEN Saclay–CERN–EPFL–
Geneva University–GSI–Thomson TMS

J.-C. Brisson[3], E. Bonomi[5], P. Borgeaud[3], J.-J. Dumont[5], J. Feyt[4], M.-N. Gaujour[8], B.W. Kolb[7], A. Krikelis[1], M. Kunt[5], J.-P. Lamarcq[8], R.M. Lea[1,2], M.B. Martin[6], T. Reed[5], F. Rohrbach[4], A. Sandoval[7] and M. Tomassini[5]

1 ASPEX-Microsystems Ltd, United-Kingdom
2 University of Brunel, United-Kingdom
3 CEA–CEN Saclay, DPHPE/SEIPE, France
4 CERN, European Organization for Nuclear Research, Geneva, Switzerland
5 EPFL, Ecole Polytechnique de Lausanne, Switzerland
6 Geneva University, Faculté des Sciences, Geneva, Switzerland
7 GSI Darmstadt, Germany
8 Thomson-TMS, France

ABSTRACT

The need for a dramatic increase in computing power is already being felt in several fields of research. It is expected to become a major issue in the near future. In high–energy hadron collider physics, many experiments being planned for the next decade will require a large increase in their data taking and analysis capabilities. In relativistic heavy–ions physics, the computing–power bottleneck has already occurred. The amount of data acquired per event, the event complexity, and the event rate are at present increasing by orders of magnitude. In many other fields, such as image–processing, cellular automata, neural networks, plasma physics, high–definition TV, autonomous guiding vehicles, artificial intelligence, etc., the computational power required is larger than that available from supercomputers. This new need in computational power will require the implementation of massively parallel processing architectures as the only possible solution at present in order to make the required breakthrough. The Massively Parallel Processing Collaboration (MPPC) intends to develop and exploit, in a two-year project, the new concept of associative parallel processing by combining all the efforts, know–how, and competence of the participating Universities, Research Institutes, and Industry to establish two powerful development 'platforms' in massively parallel processing and computing, one for the hardware and one for the software. As a 'Pilot Project' the Collaboration has selected the development of a fast, intelligent, image-intensified Charge Coupled Device (CCD) camera interfaced to an Associative String Processor (ASP) with 65,536 Associative Processing Elements (APEs). With this system, we want to demonstrate, in an experiment at CERN, the real–time data acquisition and on-line data analysis of images with 1 Mbyte of information per event. Associative parallel processing algorithms will be developed for track reconstruction of very complicated events and for large Ring-Imaging Cherenkov area (RICH) detector pattern reconstruction. The experience and knowledge gained in the realization of the Pilot Project will be applied to further problems in high-energy physics and in other fields.

Spokesman: F. Rohrbach

5002G/FR/ed

(*) The five Annexes are available upon request at the CERN/EF typing pool.

OVERVIEW

The MPPC project started in March 1989 when it was realized that the latest technological novelties in various fields were arriving at the same time to offer the possibility of developing a new fast processing system for massive amounts of data.

The future need for detectors in order to cope with the severe and stringent working conditions expected for the next generation of hadron accelerators at CERN has triggered, over the last few years, a large amount of Research and Development (R & D) for studying new ideas and new detectors with the help and support of many interested Universities, Research Institutes, and Industry (the Lepton Asymmetry Analyser (LAA) project [1,2]: a large CERN R & D programme for experimental physics with multiTeV hadron accelerators).

It so happened that, at the same time as CERN, EPFL in Lausanne and CEA–CEN Saclay in France had similar motivations to enter the new field of massively parallel computing as a possible way of providing the answer to unsolved difficulties: the demand from the physicists and engineers for increasing – by orders of magnitude – the amount of data to be analysed and, simultaneously, the requirement of speeding up the data processing above the limits of all conventional Von Neuman existing computers. This need was due to the entering of a new era of research and technology: future multi–TeV hadron colliders and very heavy ions beams in High–Energy Physics, Image Processing, and data compaction at video rates for robotics and high–definition television (HDTV), very large sampling simulation for modelling, huge data base treatment and communication, etc.

By then, at the University of Brunel (UK), the ASPEX-Microsystems company was developing a revolutionary parallel architecture for making a breakthrough in computing power. This company is developing a new device: a string of Associative Parallel Processors (ASPs) [3]. It offers a potential breakthrough towards Tera Operations Per Second (TOPS) performance at a reasonable price (target price 1 MOPS/$) with flexibility, scalability, and efficiency. A machine based on ASP, called TRAX-1, is under construction for the NA35 experiment at CERN [4] . This machine will have 16 kASP elements. This project is supported by the NA35 Collaboration and the LAA project. Dedicated parallel software is also being presently developed and an important effort is being devoted by CERN and NA35 to the application of TRAX-1 in view of speeding up, by orders of magnitude, an existing conventional image workstation installed in Frankfurt for processing streamer chamber pictures of events coming from the NA35 heavy ion experiment at CERN.

At the same time, the development of large–area CCD detectors with high gain and granularity is also in progress in Industry [5]. Thomson-CMS can now offer a one million pixel CCD sensor with low noise, fast clear, antiblooming protection, and equipped with a 25 ms parallel read-out. Development work is going on towards higher reading frequencies, better light response, a larger number of pixels, etc.

The MPPC project proposes the development (hardware and software) of a massively parallel processing system, based on the ASP, for on–line analysis of the high–energy physics experiments at CERN. In addition, the project will demonstrate and evaluate the system's capabilities for other applications of high performance processing requirements (e.g. numerical simulation of physical phenomena, Cellular Automata, Neural Networks, Aritificial Intelligence (AI), etc.).

A software platform will be organized at the Ecole Polytechnique Fédérale, Lausanne (EPFL) with the help of ASPEX for the highly specialized ASP software. In order to develop intelligent ASP algorithms, a totally new approach is necessary for programming, which implies an important effort to change the way of thinking: there is no more addressing and the massively parallel processing is done inside the memory depending on the contents of the data and activity registers, with a minimum interaction with the outside world, and the operations are mostly conducted at the bit level. This large support in software will be essential for the proposed Collaboration.

A strong and complementary hardware platform will be launched by Saclay together with ASPEX to develop and construct the new modules.

Demonstration and evaluation of the new techniques will be performed at CERN in real applications. For this purpose, a Pilot Project will be established by the Collaboration: it consists in defining, designing, constructing and demonstrating a fast, intelligent 1 megapixel CCD camera, interfaced to an ASP with 65,536 APEs and equipped with an image intensifier. Based on these results wider applications employing 10^4 to 10^6 associative parallel processors can be planned.

PROJECT PROPOSAL

Synoptic presentation

Submitted to: CERN

Title: The Massively Parallel Processing Collaboration

Project Leader: Dr. F. Rohrbach
EF Division
CERN
1211 Genève 23
Telephone: (022) 767 4149
Telefax: (022) 783 0600

Collaborators:

Institution	Responsible
ASPEX	Prof. R. M. Lea
Brunel University	Prof. R. M. Lea
CERN	Dr. F. Rohrbach/EF
EPFL	Prof. M. Kunt /LTS
GSI	Dr. A. Sandoval/NA35
Geneva University	Prof. M.B. Martin/DPNC
Saclay	Dr. P. Borgeaud/DPHPE-SEIPE
Thomson-CMS	Mr. J.-P. Lamarcq

Platforms

SOFTWARE	EPFL	Dr. J.-J. Dumont /SIC
HARDWARE	CEA–CEN Saclay	Mr. J.-C.Brisson/DPHPE-SEIPE
CCD	CERN	Mr. J. Feyt / EF

Time scale:

Phase 0:	Oct.	1989	Apr.	1990	Foundation
Phase 1:	Apr.	1990 -	Nov.	1990	Definition
Phase 2:	Dec.	1990 -	Sep.	1991	Construction
Phase 3:	Oct.	1991 -	Apr.	1992	Demonstration

Funding requirements:

Phase 0:	110 kSF
Phase 1:	400 kSF
Phase 2:	925 kSF
Phase 3:	200 kSF

Date: 27 July 1989

1. PERSPECTIVE

For nearly three decades, the elusive target of an extremely powerful super-computer has stimulated the development and evaluation of parallel computer architectures. In the 1960s and 1970s, pioneer machines (e.g. Solomon, Illiac IV, Staran, CLIP, DAP) demonstrated the evolving parallel processing concepts and in the 1980s first-generation multiprocessors (e.g. Alliant FX8, Sequent Balance, Butterfly and Transporter systems), array processors (e.g. MPP, DAP and GAPP) and hypercube architectures (e.g. NCube and Connections Machine) have become commercially available; each claims impressive performance for mainly ad hoc benchmarks. Indeed, many authorities are now proclaiming the age of very powerful computer–, graphics-, and database-servers, based on Massively Parallel Processors (MPPs), for networked CAD/CAM, military, scientific, medical, and office applications.

However, although first-generation MPPs have achieved significant increases in processing performance, they have not yet demonstrated a sufficiently significant breakthrough in application flexibility and cost-effectiveness to be commercially successful.

First-generation MPPs are highly machine-oriented, each being dominated (in cost and performance) by the particular inter-processor communication network (e.g. pipeline, mesh, hypercube, tree, pyramid, Omega, Banyan, etc.) favoured value, but may have born little relation to real (problem-solving) application requirements.

1.1 Real-time Massively Parallel Processors (MPPs)

In contrast to the first-generation parallel computer experience, an important application class for second-generation MPPs will include real-time signal and data–processing systems operating in real-world locations. Achieving performance levels up to 1 TOPS and supporting continuous data input, such general-purpose (problem-solving) computing systems will attempt the rapid detection, analysis, and decision-making (based on recognition or understanding) of object-related information and the determination of some appropriate response, as indicated in Fig. 1.

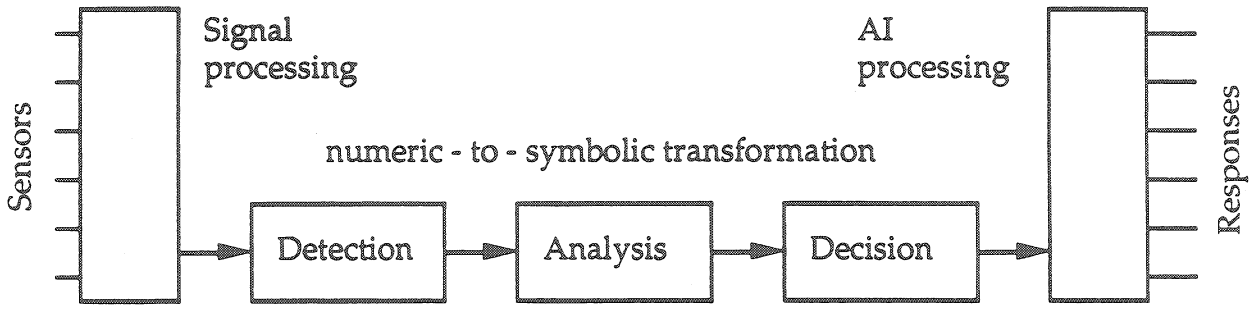


Fig. 1 Real-time signal and data–processing system

Sensors may include any form of acoustic, visual, thermal, mechanical, electromagnetic, or nuclear–radiation energy, and applications include detector readout, remote sensing, surveillance, and tracking in high–energy physics, aerospace, seismological, and in biomedical environments. In particular, real-time computer vision systems suggest ideal massively parallel processing application examples, with a processor allocated to each pixel of the image to be processed.

Typical detection tasks range from signal reconstruction of restoration (to compensate for sensor non-uniformities, ageing, and eventual failure), through filtering (to improve signal-to-noise ratios and enhance particular signal features) to correlation (to discriminate relevant signal information from background clutter).

Typical analysis tasks range from grouping data associated with meaningful objects, through specific transforms for feature extraction and edge or track following (to establish feature connectivity) to object quantification (for length or area measurement or centre-of-mass coordination) and listing of relevant object properties.

Typical decision tasks range from pattern matching (correlation of invariant object descriptors with model descriptions) and navigation of semantic networks through hypothesis analysis to interpretation of object-related information to facilitate decision-making and response determination.

Real-time signal and data processing applications are characterized by the need to provide a rapid response to information derived from a high-speed continuous data stream. For example, real-time computer vision with non-interlaced (60 Hz) raster-scanned image frames requires the completion of all processing tasks (including data input and output) within a frame-time of 16.7 ms, suggesting processing rates of 10–1000 GOPS (GigaOPS) and input data rates approaching 1 Gbytes/s. Clearly, such high performance can only be achieved with massively parallel processing.

However, such end-to-end processing (namely problem-solving) systems encompass a sequence of tasks, from signal processing through some form of numeric-to-symbolic date conversion (usually achieving considerable data reduction) to AI (Artificial Intelligence) processing. Indeed, processing tasks are not solely confined to the low-level arithmetic operations on 1D or 2D arrays typical of first-generation parallel computing, but the wide variety of tasks also includes higher-level operations on more complex data structures (e.g. sets, tables, trees, or graphs). Moreover, the nature of these application environments ensures dynamically changing functionality requirements as sensor technology and application understanding improve.

Consequently, second-generation MPPs must satisfy significantly different application/ design requirements from their first-generation predecessors.

1.2 Need for massively parallel processing applied to high–energy physics

There are essentially two kinds of experiment at CERN: **Fixed–target physics and Collider physics**. The topology for the detectors is different in the two cases (Fig. 2).

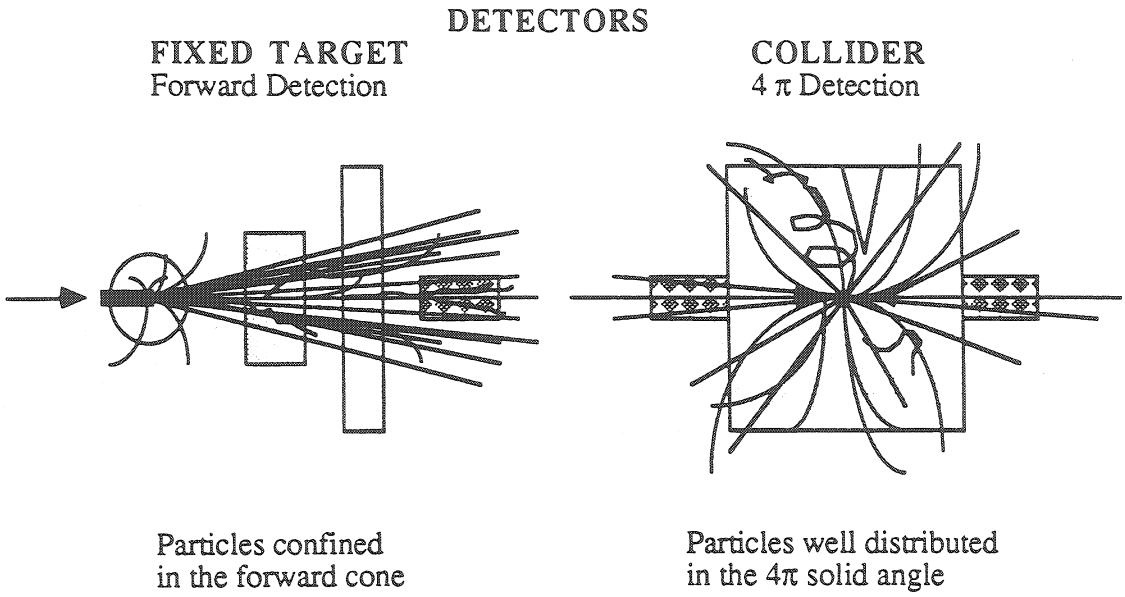


Fig. 2 Topology of high–energy experiments at CERN

However, the detecting principle remains the same: starting from the vertex of the interaction a succession of tracking and ionization detectors, calorimeters and velocity counters, muon tagging and tracking devices are disposed in an efficient way in order to extract all the possible information from the interaction products. The simultaneous knowledge of the totality of the information coming from all the detectors surrounding the collision point is necessary for the evaluation of the event and, consequently, there exists a natural need for processing all the data in a parallel way and not sequentially (Fig. 3).

Except for imaging detectors, the number of pixels of information coming from all the detectors has up to now been limited to 10 to 100 kbytes and the event rate was not greater than 100 kHz, the interesting events being stored at a rate of a few Hz. A conventional approach using sequential machines was able to cope with these requirements by a logical implementation of some parallel treatment of the information for triggering the data acquisition out of a pipelining pre-storage system. However, the data analysis itself, which is done off-line, is taking a very long time compared to the running time of the experiment, giving rise to an imbalance between the duration of the experiment and the physics outcome. Typically years of analysis can be generated by weeks of data taking. Moreover the feedback from the experimental results which could be used to control and improve the data taking is

coming by far too late and often the experiment must face a cruel issue: the data are not usable for the analysis as expected.

For the case of imaging detectors, the number of pixels of information per event is so high (many Mbytes) that, up to now, film has had to be used as storage support and a very long off-line analysis has had to be done (for example, 10 h on an image workstation are necessary for the analysis of every event in the NA35 heavy–ion physics experiment).

In order to improve the redundancy in particle detection - higher and higher numbers of tracks per event - and the quality of the physics analysis – mass, momentum and energy resolution – it is necessary to increase the number of pixels of information per track for tracking detectors, to reach a higher granularity for calorimeters and to provide more pads for RICH counters. This means that the total number of pixels per event is going to increase.

At the same time there is a deep physics interest to search for rarer phenomena and higher mass particles. In order to proceed along these lines, there is a need to increase both the luminosity of the accelerators and their energy. The main parameters of existing and future hadron accelerators show the huge step which is expected to take place in the rate of events and charged multiplicities for hadron colliders (Table 1).

TABLE 1
Hadron accelerators: main parameters for detector

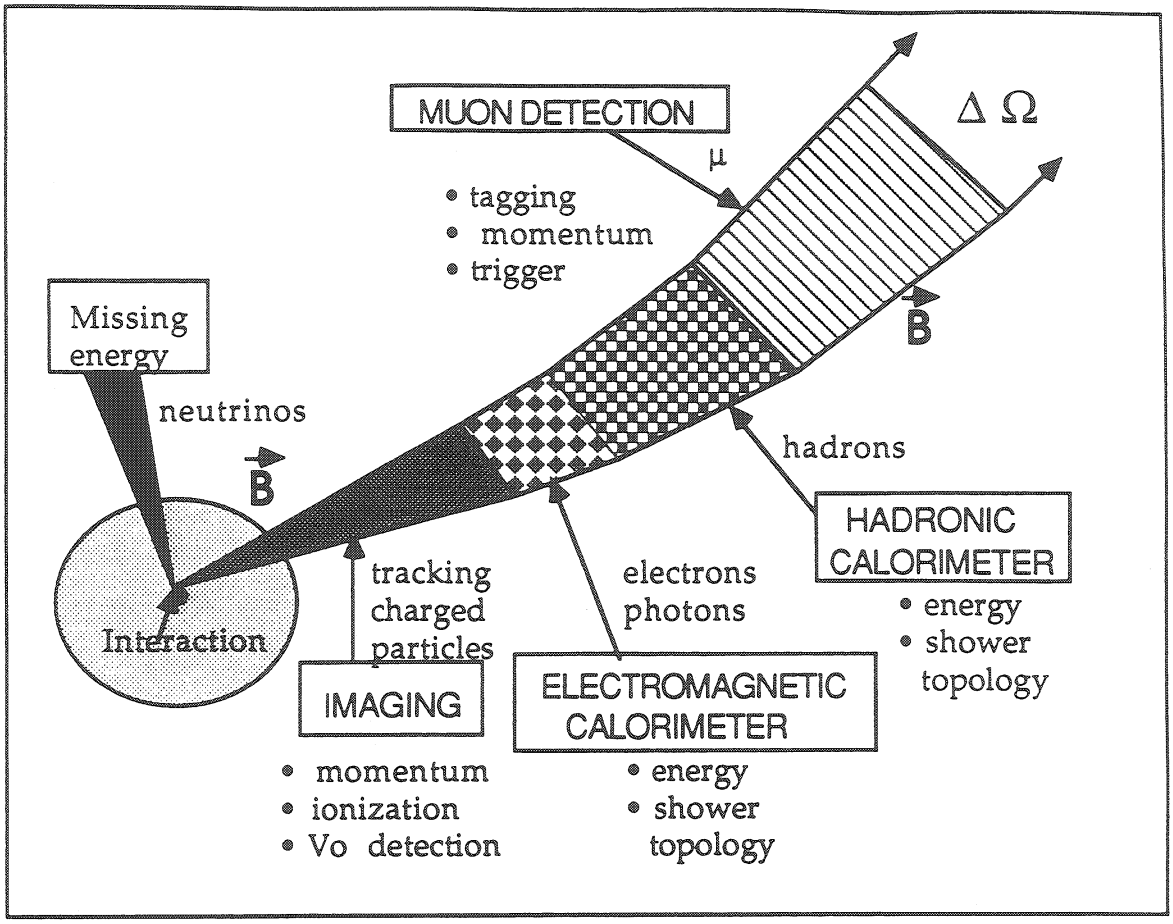
Machine	Particles	$\sqrt{s}$	L	Δt	R	<n>
SPS $p\bar{p}$ (CERN)	$p\bar{p}$	0.63	3×10^{30}	3.8 μs	~ 200 kHz	30
Tevatron (Fermilab)	$p\bar{p}$	1	10^{30}	3.5 μs	~ 100 kHz	35
UNK (URSS)	pp	6	10^{32}	6 ns	~ 10 MHz	60
LHC (CERN)	pp	16	10^{33}-5×10^{34}	25-5 ns	up to ~ 5 GHz	80
SSC (Texas)	pp	40	10^{33}	18 ns	≈ 100 MHz	104
ELOISATRON (Italy)	pp	100	$\approx 10^{33}$	25 ns	≈ 100 MHz	134

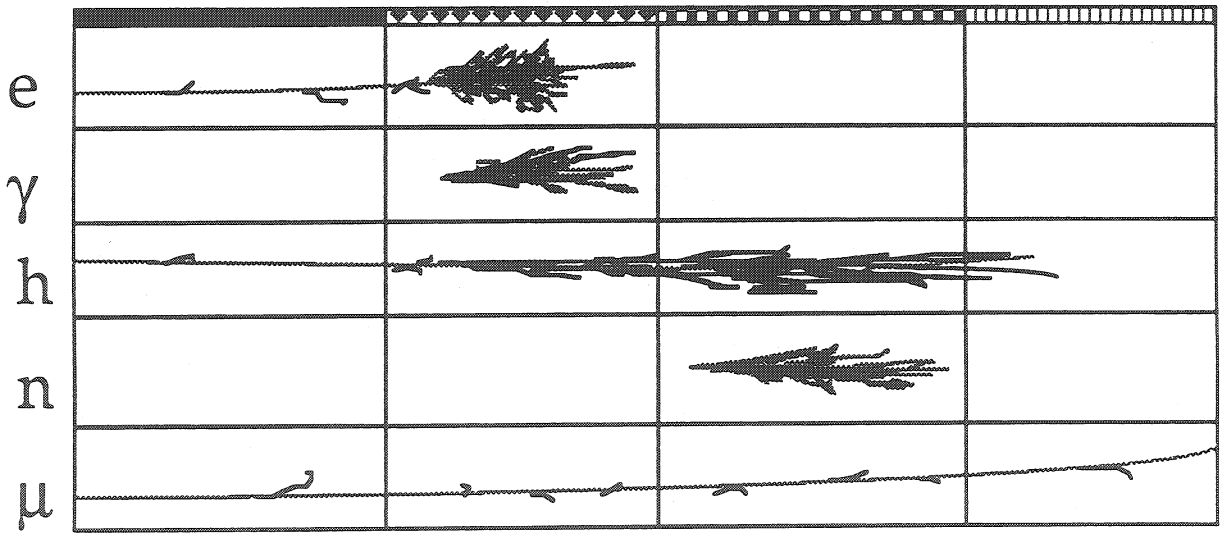
with: $R = \langle r \rangle / \Delta t = L\sigma$

where
- R Rate of interaction (s^{-1})
- <r> Number of interactions in each crossing
- Δt Time between crossings (s)
- L Luminosity ($cm^{-2}s^{-1}$)
- σ Cross section for interaction [cm^2]
- $\sqrt{s}$ cms energy (TeV)

and: $\langle n \rangle = -6.55 + 6.89\ s^{0.131}$ (power fit)

- <n> Average charged multiplicity for non single diffractive events [6]
- s c.m.s. energy squared (GeV^2)





Different kinds of particles ⇒ Different kinds of detectors :

Imaging Vertex Detector	Electromagnetic Calorimeter	Hadronic Calorimeter	Muon Tracking

Classical approach:

TPC, Streamer Chamber	Scintillators & dense High Z materials	LST, Drift Chambers

Fig. 3 Detection principle in high-energy experiments

There is another difficult challenge for detectors in ultra-relativistic heavy–ion experiments: going from light to very heavy ions, one has to face a continuous increase in the total charged particle multiplicity per event. The increase in average multiplicity for central collisions, observed or to be expected for future beams, is as follows:

1986	O^{16}	$\langle n \rangle_{central} \cong 250$
1987-91	S^{32}	$\langle n \rangle_{central} \cong 400$
≈1993	Pb^{208}	$\langle n \rangle_{central} \cong 2500!$

In most of the present fixed–target experiments, the data–acquisition rate is only limited by the detector and the electronics but not by the beam and target from which very high luminosities could be obtained.

Consequently, for colliders as for fixed–target experiments, one expects not only a drastic increase in the rate of the events but also an increase in the number of particles filling the detectors for every interaction. Rates are increased by ~ 3 orders of magnitude, the time between successive events is decreasing down to a few ns and the number of bytes per event has to increase to many Mbytes in order to disentangle the interaction products.

At present, there are no solutions to these difficulties. However, there is a clear line of research and development which could offer the necessary breakthrough: the use of massively parallel processing in order to gate the data flow and to analyse on-line the events (tracking, clustering, event checking, etc.).

1.3 Need for massively parallel processing applied to other sciences

It is not only in high–energy physics that there exists a need and motivation in the direction of massively parallel processing. In many fields of science and technology, existing algorithms require huge computational resources to yield useful results. Most of them are in fact rather simple, but imply repeating the same set of operations on large amounts of data. This motivated the development of the vector processors of the so-called 'supercomputers', which are now dominating the market, but mostly by their price. It is now clear that this very expensive approach is still extremely limited and does not satisfy the present users of the 'supermachines'. It seems that the only way out of the computing crisis is to be found in massively parallel processing.

Let us summarize what are the main kinds of applications that would benefit from parallel machines, and what are their problems with the existing ones.

1.3.1 Numerical simulation of physical phenomena

Traditionally, such simulations use a mathematical model consisting of differential equations defined on a continuum. An algorithm, or numerical model, is used to discretize

and numerically solve the equations. Classical applications of this approach are, for instance, structural and mechanical engineering, fluid dynamics, and other physical and chemical processes such as combustion or solidification.

For this kind of problem, parallel architectures belonging to the MIMD (Multiple Instructions Multiple Data) class can be of interest: relatively cheap but powerful processors are assembled, each one executing a different stream of instructions on data stored in a local memory. Physical connections between processing elements can be with nearest neighbours, as for the Inmos transputers, or following a hypercube scheme, as for the Intel machines. However, in both cases the Minsky conjecture, which states that the effective power of a machine goes like log N if N is the number of processors, turns out to be a good approximation in most practical cases. This disappointing limitation stems from the overhead implied by the communication protocols between processors, and also from the control operations, which are sequential by nature.

This somewhat reduces the interest of transputers, which were once thought to be ideal solutions in the realm of numerical simulation and floating–point data processing. Indeed, the user must create his own communication protocols, tailored to his special application, which excludes any kind of efficient FORTRAN programming for instance. Specialized operating systems (Helios) or FORTRAN compilers (Meiko) can help, but in any case the situation remains unsatisfactory.

The Intel hypercubes are supposed to be in a better situation as far as the communication problems are concerned, as they can rely on much better software tools and on a theoretically more sophisticated connection scheme. However, good efficiencies can still only be reached for very few real life applications, and load balancing of the various processors, as for any MIMD architecture, remains a critical and largely unsolved problem.

1.3.2 Cellular automata

It is an alternative way of modelling and simulating physical phenomena: space is discretized by a lattice whose grid points (site) are allowed a finite number of values (states). The state of each site evolves step by step in the model, following local microscopic laws (rules) which, if chosen properly, produce the macroscopic behaviour to be simulated. The intrinsic granularity of cellular automata algorithms leads naturally to their implementation on massively parallel machines. They are very inefficient on classical number crunchers, as they need only a few bits per operation.

1.3.3 Neural networks

The brain is the most massively parallel processor existing today. It seems like a good idea to try to exploit its basic features for the design of electronic processors, if they are to

solve by analogy he same kind of problems as the brain does, namely associative memory storage, learning, pattern recognition, control or optimization. The bulk of neural algorithms consists in summing weighted quantities.

The main problem of these specialized processors is that they can only run one hard-coded algorithm, and can therefore be only used for one type of application. It also implies that the behaviour of the chosen algorithm in various situations must first be checked using simulators, before building the processor. This can only be done efficiently on other parallel machines. If the simulation is fast enough, the specialized processor may become unnecessary.

1.3.4 Artificial Intelligence

Today, it seems unlikely that large neural networks could really work like an intelligent brain and solve efficiently any AI problem. However, there are many other techniques of AI which also involve parallel processing. For instance, the 'assumption-based truth maintenance' algorithm was implemented in a parallel fashion by Xerox, with the aim of applying it to robots. In that case, it would need compact and cheap MPPs. Another example is the 'case-based reasoning', in which a huge amount of cases are to be considered, preferably in a parallel way. There is a large variety of expert systems which would benefit from such algorithms, or from other ones using various kinds of active objects, structures, sets, and graphs. No existing machine can easily and efficiently allow for the variety of data structures appearing in this kind of application.

1.3.5 Other applications

Without going into details, let us also mention briefly the need for parallelism in operational research (mostly for ordering problems), graphics rendering (for instance, ray–tracing algorithm for the generation of realistic pictures), and particularly image or signal processing in astronomy, geophysics, medicine,.etc.

1.4 Second-generation massively parallel processing requirements

To be cost-effective as second-generation MPPs, parallel computer architectures will need to satisfy both architectural and engineering requirements, as discussed in the following sub-sections.

1.4.1 Architectural requirements

(i) Application flexibility

In order to recover high development costs, second-generation MPPs must serve a variety of applications and users; 50 users will need to amortize high procurement costs

through a wide range of both numerical and non-numerical information processing applications.

Moreover, real (i.e. problem-solving) computer applications, as opposed to first-generation MPP benchmarks, usually involve a range of different data stuctures and different operational requirements across quite a wide span of computational tasks. For example, computer vision comprises a wide range of data structures (e.g. sets, arrays, tables, trees, and graphs) encountered in iconic-to-symbolic image processing and subsequent image understanding.

By allocating a processor to each node of a data structure and by linking the processors in order to emulate the relationship existing between such nodes, application parallelism can be exploited with the architectural parallelism of multi-node processing and multi-link navigation.

However, most first-generation MPPs are dedicated to particular data structures and optimized for particular computational tasks. Continuing with the computer vision example, mesh architectures have been applied, with reasonable success at the pixel (iconic) level, but this solution may only serve to transfer the computational bottleneck to a higher level of processing. Similarly, tree architectures perform better for (tree-structured) symbolic data processing, but are not well-suited to lower level processing.

In summary, application flexibility (maintaining efficient support of different data structures) will be an important requirement for second-generation MPPs.

(ii) ***Architectural scalability***

The number of processors in first-generation MPPs tends to be fixed or is severely restricted (e.g. 1 k, 4 k, 16 k or 64 k), whereas problems are not so conveniently sized. Moreover, users will wish to extend existing MPPs to achieve more processing power. Consequently, architectural extensibility (i.e. scaling of parallel processing power) will be a key issue for second-generation MPP design.

(iii) ***Computational efficiency***

First-generation MPPs claim up to order N (i.e. O[N]) improvement in performance for O[N] parallelism. However, although this may be possible for certain specific (highly parallel) processing tasks, other tasks may not benefit from such high degrees of parallelism. In fact, for real (i.e. multitask) applications, the overall performance of such MPPs is more likely to be worse than O[logN] (namely Minsky's conjecture). Indeed, the somewhat reluctant realization that emerges from the analysis of first-generation programs is that their performance is not affected so much by the potential processing power of the processor ensemble, but more by the loss of efficiency due to poor processor utilisation and the hidden

processing overheads associated with the strategies adopted for parallel program control, as discussed below.

(a) sequential processing overheads, associated with the input and output of (vector) data values;

(b) sequential processing allocation, resulting in redundant sequential processing and low concurrency within the processor ensemble and inter-processor communication network, due to a mismatch between the applied parallelism of the computer and the natural (data-level) parallelism of the data structure being processed;

(c) sequential processing overheads, associated with the initialization and scheduling of the concurrent execution of mutually exclusive processes/operations, resulting in inefficient exploitation of natural (control-level) parallelism in the algorithmic requirement;

(d) sequential processing due to requirements for the storage and processing of scalar (as opposed to vector), data (e.g. during the calculation of intermediate results);

(e) parallel processing overheads, due to the initialization and synchronization of inter-processor communication (which would not be suffered by a sequential processor: for example, navigating irregular tree-structured data on an orthogonal array processor could entail much routing through redundant processors with significant control and communication overheads.

Requirements for maximal computational efficiency are outlined below in (iv) through (ix).

(iv) ***Pipelining data input***

In order to support on-the-fly processing of continuous input data streams, at rates up to 1 Gbytes/s, and avoid associated sequential processing overheads, real-time MPP architectures must adopt an efficient pipelining policy for data input.

(v) ***Inter-processor communication network configurability***

Optimal processor utilization is achieved when the MPP's network topology exactly matches the particular data structure being processed and, inevitably, the MPP would suffer loss of computational efficiency for non-matching data structures. Thus, for flexible matching of the applied parallelism of a general-purpose MPP with the natural (data-level) parallelism of different applications and for minimization of the parallel processing overheads mentioned, a reconfigurable processor interconnection strategy is required.

Consequently, the degree of network configurability (to exploit data-level paralllelism) will be a most important parameter for the computational efficiency for second-generation MPPs.

(vi) Overlapping sequential and parallel processing

In order to avoid loss of computational efficiency due to the sequential processing overheads discussed above, it is essential that all sequential processing is overlapped with parallel processing.

(vii) Programmability

Apart from systolic arrays, most first-generation MPPs are programmable. Similarly, second-generation MPPs will require programmability in order to gain application flexibility and achieve task-oriented specialization cost-effectively. Indeed, for flexible control over pipelining data input, inter-processor communication network configurability, process scheduling, and overlapping sequential processing as discussed above, programmability is an essential requirement for the optimization of computational efficiency.

1.4.2 Engineering requirements

Second-generation MPP applications are likely to dictate an engineering approach (as opposed to a computer–science approach) to parallel computer design. For example, real-time parallel computer systems must operate reliably within environmental restrictions, taking into account maintenance schedules and cost limitations. Such engineering issues are discussed below.

(i) Environmental restrictions

Certain MPP application environments (e.g. high–energy physics detector front ends, space, avionics, underwater and automotive) present the most formidable design challenge by severely restricting size and weight, assembly and packaging techniques, and power supply and dissipation, as discusssed below.

(a) Size and weight

Typical cavities for the above application environments measure significantly less than 1 cubic foot and weight reduction can be a key issue. Rather less demanding, but nonetheless important, are the size and weight requirements for workstations, which must fit acceptable (e.g. under desk) locations of, say, less than 5 cubic feet.

In contrast, the 65,536-processor Connection Machine, representing the state-of-the-art in first-generation MPP compaction, occupies 75.4 cubic feet. Clearly, second-generation MPP applications will require a step-function increase in the density of microelectronics

assembly and packaging to achieve the required reduction in the size and weight of first-generation MPPs.

Consequently, the trend towards High-Density Multi-layer Interconnect (HDMI) technologies [i.e. highly-compact multi-chip modules, constructed by mounting unpackaged VLSI chips on alumina or co-fired ceramic substrates or even silicon wafers (i.e. hybrid wafer-scale integration)] is of significant interest for size and weight reduction.

(b) Assembly and packaging

The application environments mentioned above often present a most formidable assembly and packaging challenge, owing to harsh operational conditions (e.g. extreme temperature range, radiation, electrical noise, mechanical vibration, humidity or even a corrosive atmosphere), which further stimulates interest in HDMI–VLSI multi-chip modules.

(c) Power supply and dissipation

The application environments mentioned above typically restrict supply and thermal design with power limits of less than 1 kW.

In contrast, the CM-1 and CM-2 versions of the 65,536-processor Connection Machine, dissipate 12 kW and 28 kW respectively. Thus, even in an office environment, the power dissipation of such an MPP workstation would be unacceptable. Clearly, second-generation MPP applications will also require a step–function reduction in the power consumption of first-generation MPPs.

(ii) Reliability and maintenance

The reliability and ease of maintenance of relatively expensive machines, such as MPPs, will always be important issues. In particular, for the reasons given below, highly-compact fault-tolerant replaceable multi-chip hybrid modules, are recommended for MPP implementation.

(a) Fault-tolerance

Second-generation MPPs would offer improved system reliability and ease of maintenance, by adding redundancy to the processor ensemble and achieving in-service fault-tolerance with self-testing and electronic reconfiguration to isolate faulty processors. Moreover, fault-tolerant second-generation MPPs could support graceful degradation or even fail-safe systems.

(b) Modular systems

With service lifetimes extending beyond 10 years, ease of maintenance will be a key issue for second-generation MPPs, which will stimulate a trend towards modular parallel

computing systems. Indeed, ease of maintenance would be greatly improved by the use of line replaceable modules, since simple plug-in module replacement would provide the opportunity of minimizing down-time and maximizing availability.

(c) Microelectronics and packaging technologies

Increasing micro-miniaturization offers improvements in reliability and ease of maintenance, owing to the integration of chip-to-board-to-chip interconnections, the consequent reduction of failure-prone board connectors, and the increasing effectiveness of plug-in module replacement. Consequently, the trend towards the implementation of second-generation MPPs with HDMI-VLSI multi-chip modules, recommended above for size and weight reduction, is reinforced by these advantages.

(iii) Cost–lifetime

The history of maxicomputer, minicomputer and microcomputer development suggests that first-generation MPPs will rapidly become obsolete as inevitable advances in microelectronics technology (e.g. increasing chip size and reducing feature size) allow more exciting products to enter the market place. Thus, second-generation MPP customers are likely to be wary of technological change. Indeed, wise customers will be looking for security and growth potential in 'future-proof' investments in MPP technology.

With service lifetimes extending beyond 10 years, architectural and software durability under functional upgrade (as application requirements change) and technological upgrade (without modification of existing software) will be very important requirements for second-generation MPPs. Indeed, the trend towards modular systems (see above) could provide some degree of system (design) stability and technology independence, with simple replacement of MPP modules.

(iv) Cost–performance

Emerging from the first-generation MPP experience is the realization that the cost–performance of second-generation MPPs will not be dominated by the exciting increase in processing power, but by the much more mundane details of the implementation cost of the processor ensemble and, especially, the inter-processor communication network, plus the peripheral costs of the global memory, control unit, and data interfaces.

Indeed, significant increases in processing performance have enabled first-generation MPPs to stimulate commercial interest in parallel computing; applications exist, parallel algorithms are being developed, and architectural principles have been established. However, first-generation MPPs are unlikely to break through the cost barrier currently impeding commercial exploitation. For example, first-generation MPPs claim up to

1MOPS/$1K, which represents only O[1] improvement in cost–performance for O[N] parallelism.

In contrast, second-generation MPPs must aim at O[2]–O[3] improvement, in order to provide an adequate return on the high capital investment required to install new hardware, for programmer retraining and, especially, for new software development. Hence, 1 MOPS/$ represents a realistic cost–performance target.

In view of the dominating influence of implementation cost and the inevitability of processor redundancy (see below), it follows that second-generation MPPs should be specifically designed to exploit the economic opportunities of microelectronics technology.

(v) Silicon-efficient chip architecture

Implementing MPP architectures in silicon introduces a different set of cost–performance trade-offs from those previously encountered by first generation MPP designers; these being rather more cost-reduction than performance-enhancement oriented. In contrast to first-generation MPP developments, top–down design of MPP architectures (designed without detailed regard to microelectronic opportunities and constraints) is likely to be less cost-effective than bottom–up design of MPP chips (without regard to architectural requirements).

An important lesson can be learned from the history of computer memory technology. Random-access memories (RAMs) incorporate massive redundancy; since, except for the addressed storage location, all other locations are inactive. Consequently, early computer memories, based on expensive implementation technologies, were kept small to maintain cost-efficiency. However, as technology improved, minimizing memory implementation cost, memory is now so cheap that even modest computers can boast a large main store. Clearly, advances in microelectronics technology have reduced the memory redundancy problem to insignificance. Thus, the challenge is to apply a similar technological trend to improve the cost-effectiveness of the processor ensemble.

With this challenge in mind, it is significant that although memory locations have been greatly reduced in size, memory redundancy remains constantly close to maximum. In contrast, the much lower level of processor redundancy tends to compensate for higher processor complexity.

A concerted attempt to render second-generation MPPs as cost-effective as RAMs would be impelled by the following design attempts to achieve silicon-efficient MPP chip architectures.

(a) Maximize the packing density of the processor ensemble

In order to reduce the cost of inevitable processor redundancy to a negligible level, this objective requires a maximum number of processors which can be packed on an economical silicon die; this entails minimization of processor layout area to benefit from a high intrinsic chip yield.

(b) Incorporate the inter-processor communicating network on chip

This objective entails the minimization of the geometrical layout of the inter-processor communication network and supporting global routing for the processor array. As layout feature sizes continue to reduce and VLSI chip sizes increase, the inter-processor communication network progressively dominates the device architecture, such that placement of individual processors is becoming a secondary consideration. Thus, integrating the network within the processor ensemble is of major importance for the reduction of MPP implementation cost.

(c) Minimize input–output pinning overheads

This objective recognizes that the cost of microelectronic assembly is dominated by interconnection technology.

In summary, engineering barriers concerning size, weight, reliability, power consumption and cost are currently impeding progress towards the commercial exploitation of second-generation MPPs for real-time parallel computing. Indeed, it is clear that, in order to achieve a breakthrough, it is not just TOPS performance that is required, but second-generation MPPs must also achieve step-function improvements in TOPS/ft^3, TOPS/lb, TOPS/failure, GOPS/W, and MOPS/$. For example, assuming that weight and power problems can be resolved, it is generally recognized that a low-cost highly-compact fault-tolerant real-time computer vision system could launch high-volume products in the robotics market.

Current trends in microelectronics and packaging technologies toward highly-compact plug-in replaceable HDMI–VLSI multi-chip (hydrid) modules augur the desired breakthrough and, at the same time, simplify problems of tolerating harsh environmental conditions. Moreover, simple module replacement satisfies the need for ease of maintenance and provides a convenient means for both functional and technological upgrade during operational lifetime.

1.5 Dedicated pipeline architectures

An ideal real-time parallel computing system would instantaneously process each data sample on input and, consequently, the traditional approach to the implementation of such

systems is a pipeline of different processors, as shown in Fig. 4, with each stage being dedicated to a specific task which is executed on-the-fly as data flush through its processor. Clearly, in order to process a continuous stream of data samples, the task–processing rate of the first stage of the pipeline must match the maximum input data rate. However, the performance of each subsequent processor in the pipeline, can be optimized to reduce the overall latency delay of the output response.

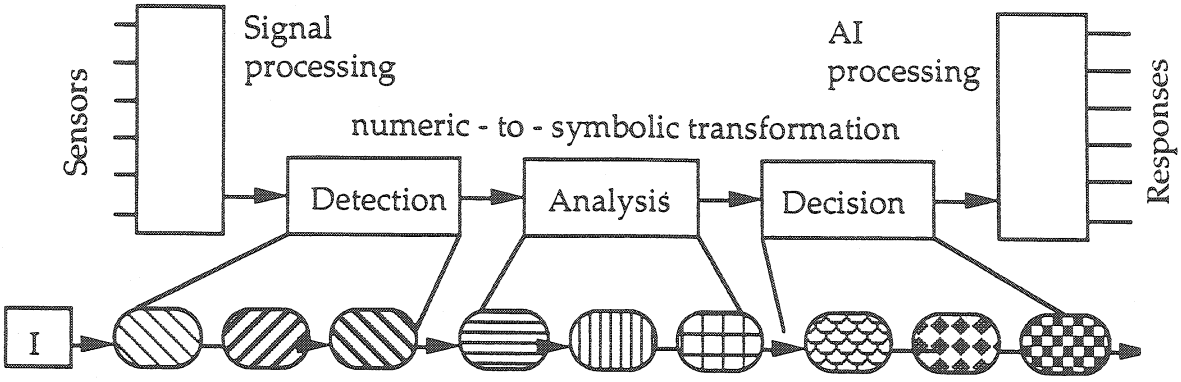


Fig. 4 Dedicated pipeline processor

Such spatial partitioning of task parallelism implies that the scaling of overall parallel processing power, to match the ever-changing (input-to-output) complexities and performance requirements of different parallel computing applications, entails adjusting the length of the pipeline with the insertion or removal of high-performance hardware stages. Indeed, although the rate of data transfer between stages reduces rapidly as computation proceeds along the pipeline, particular tasks may, nevertheless, be based on quite complex algorithms, requiring the high-speed execution of very many low-level operations.

This ad hoc optimization of different processors and the emphasis on high-speed data flow between processors restricts opportunities for microelectronics and packaging technologies to reduce the number of expensive and error-prone interconnections and, thereby, overcome the engineering problems discussed above. Indeed, development, manufacture, and maintenance costs for a range of such unique (i.e. low volume) modules will inevitably remain high (compared with large volume standard parts) and system durability will be difficult to maintain. Moreover, the expense of a redundant processor in irregular parallel computing pipelines is likely to render attempts at fault-tolerance cost-ineffective. Consequently, such machines are unlikely to achieve the desired engineering breakthrough.

The traditional approach to overcoming these problems would be with a regular parallel computing pipeline of identical stages (e.g. the linear systolic array). Real-time on-the-fly processing can still be achieved, but the output response now suffers a latency delay which depends on the length of the pipeline (i.e. the number of stages).

By balancing processing rate with input data rate, the uniform pipeline processor gains an undoubted advantage in speed for signal processing. However, lack of architectural and programming flexibility would severely reduce the efficiency of the numeric-to-symbolic data conversion process and AI processing for which the common processing stages are not so well suited.

Lastly, it is interesting to note that such homogeneous pipeline architectures derive little benefit from the experimental observations, that for most real-time parallel computing applications the numeric-to-symbolic data conversion process may reduce the output data volume by several orders of magnitude (with respect to a given input data volume), and that application requirements for processing and input data rates are normally significantly different.

1.6 MIMD/SIMD array processing architectures

Architectural alternatives to dedicated pipelines, which attempt to take advantage of the above criticisms and observations, are based on homogeneous MIMD/SIMD (Multiple/Single Instruction control of Multiple Data Streams) array processing, as indicated in Fig. 5. In this case, the numeric-to-symbolic data conversation process is achieved by first distributing data elements over the identical processors of the MIMD/SIMD architecture and then executing successive tasks on the stored data. Hence, such temporal partitioning of task parallelism must be supported by data buffering to ensure that task processing rates are independent of input and output data rates.

Since application requirements for processing and input data rates are normally significantly different, such (buffered) MIMD/SIMD architectures provide a more natural medium for application flexibility than dedicated pipelines. Moreover, it should be noted that the temporal parallelism of homogeneous MIMD/SIMD array processing architectures, in contrast to the spatial parallelism of dedicated pipeline architectures, offers much greater application flexibility with a much simpler and cheaper medium for programmability and scaling of processing power.

In addition, in order to maintain high computational efficiency with MIMD/SIMD architectures, an effective balance between input data pipelining and parallel processing power must be ensured. Moreover, all sequential (scalar) processing should be overlapped (in the MIMD/SIMD array controllers) with parallel (vector) processing in the MIMD/SIMD arrays and, whenever possible, data transfer should be achieved with external data routing networks.

2. OPPORTUNITY

In terms of application flexibility, architectural scalability and all the engineering issues discussed above, the development of computer storage has been a great success. Indeed, as the ultimately cost-effective computer component, RAM storage represents an ideal model for the development of second-generation MPPs.

Consequently, since traditional computer storage is organized with a string topology [i.e. all application data structures are mapped to a string of (groups of) words in RAM or bytes on disk], it seems a logical prospect for second-generation MPPs to be organized as a string of processors.

Thus, further consideration of the perspective presented above has led to the conclusion that a simply scalable and fault-tolerant homogeneous parallel computing hierarchy, based on MIMSIMD (Multiple-Instruction control of Multiple SIMD) processors, would offer the best opportunity for cost-effective second-generation MPPs.

According to application requirements, a string of identical MIMSIMD processors would be plugged into the control bus and Data Communications Network of an appropriate modular MPP racking system, as indicated in Fig. 5.

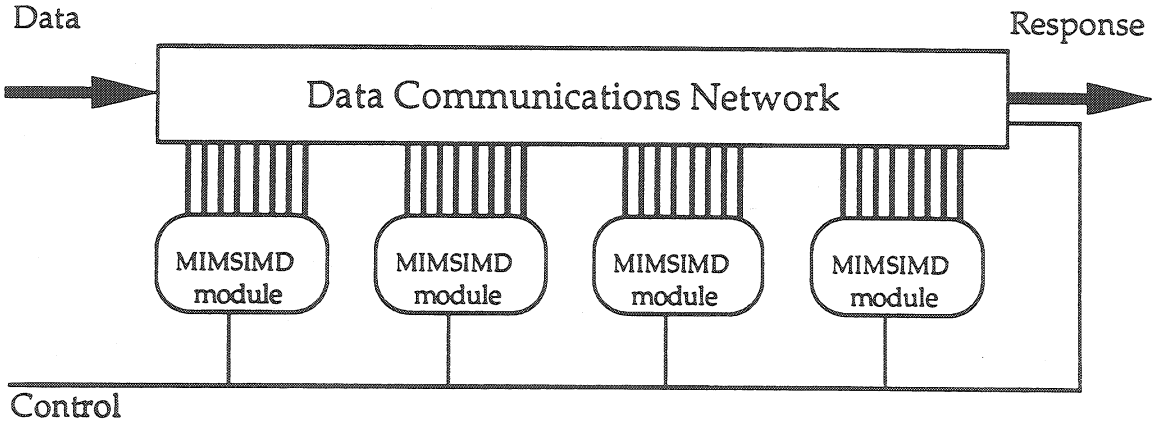


Fig. 5 Modular second-generation MPP

Each MIMSIMD processor comprises a simply scalable and fault-tolerant string of inter-communicating SIMD processors and shared Control Units (CUs), as shown in Fig. 6. The MIMSIMD processor would also incorporate a Data Interface (DI) for each SIMD processor and a common control interface, to connect to the MPP system rack.

Similarly, at the next level down in this homogeneous parallel computing hierarchy, each SIMD processor would comprise a simply scalable and fault-tolerant string of a large number of identical fine-grain processors and a reconfigurable inter-processor

communication network. As with RAM storage, the SIMD processors would be internally organized to exploit the most recent advances in state-of-the-art microelectronics.

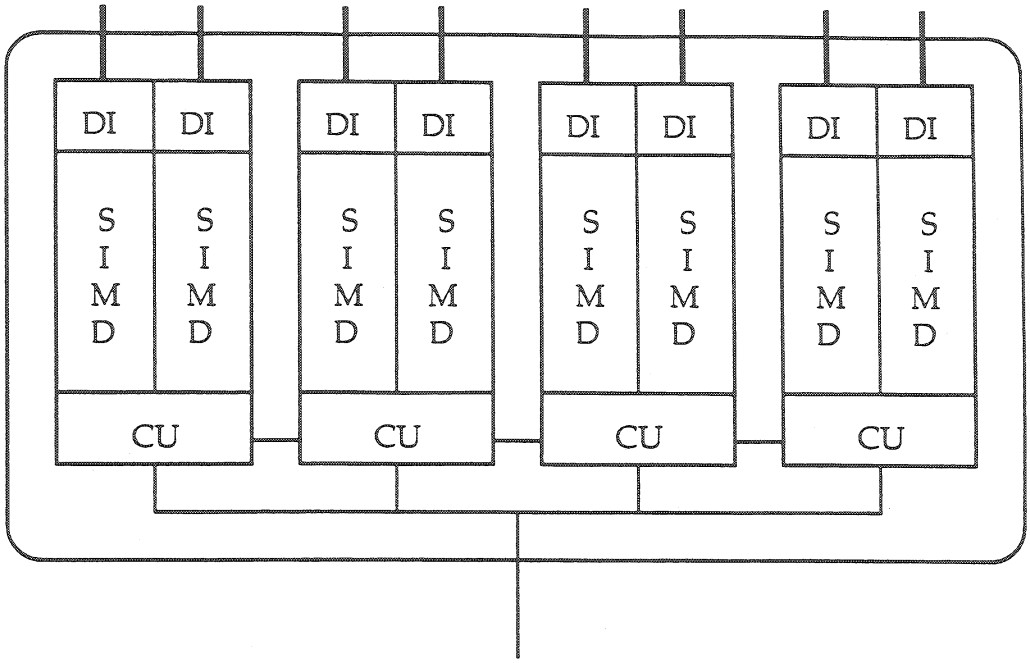


Fig. 6 MIMSIMD processor

2.1 The Associative String Processor (ASP)

Associative String Processor architecture and support software provide the basic technology for the development of versatile parallel processing building-blocks (i.e. ASP modules of the form shown in Fig. 6) for the simple construction of modular real-time MPP systems.

An ASP module comprises a MIMSIMD parallel processing structure of intercommunicating ASP substrings, each supported with an ASP Data Buffer (ADB) and an ASP Control Unit (ACU), as shown in Figs 6 and 7. The ASP module also incorporates a single Control Interface (CI) and a multiple DI, to connect to the system rack.

Based on the encouraging results emerging from research into parallel computing architecture at Brunel University and being developed by ASPEX–Microsystems Ltd., ASP technology comprises highly versatile hardware and software modules for the construction of second-generation MPPs capable of TOPS performance.

The ASP system architecture has been specifically designed to satisfy both the second-generation MPP computational requirements and the engineering constraints discussed in the perspective above. Indeed, it is claimed that the ASP is the first parallel computing architecture to achieve this design goal.

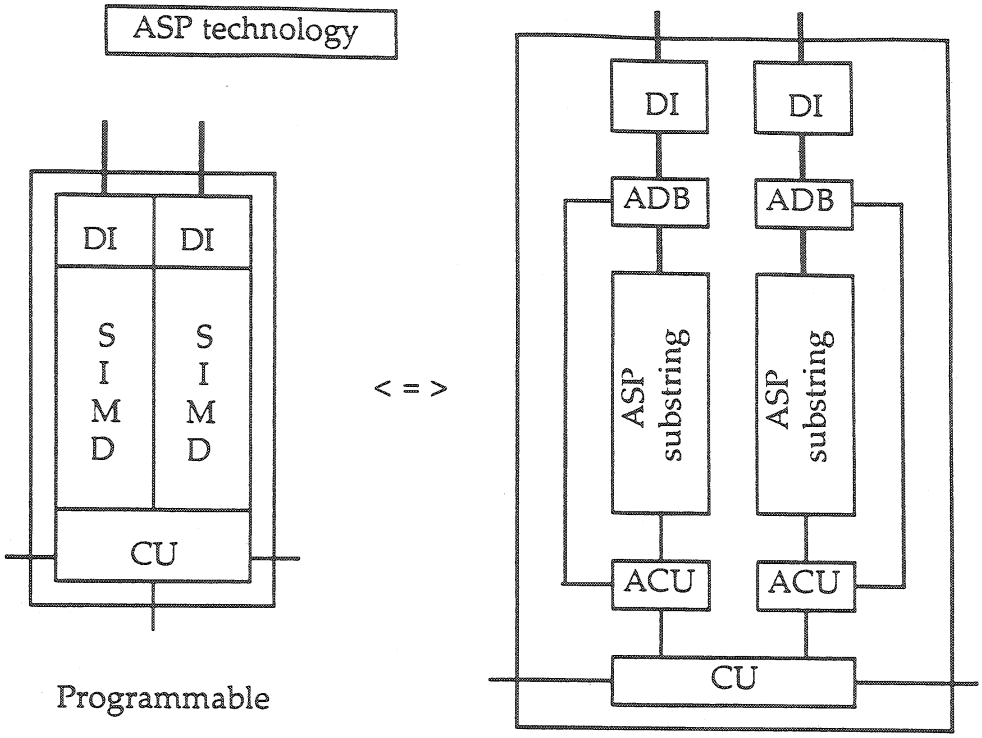


Fig. 7 ASP substrings

Based on a fully-programmable, reconfigurable and fault-tolerant homogeneous parallel processing architecture, ASP modules offer considerable application flexibility, maintaining high computational efficiency over a particularly wide range of signal and data processing applications, due to:

(i) simple configuration of ASP modules to simplify the development of MPP systems which are well matched to functional application requirements;

(ii) pipelining and overlapping input–output data transfers (via the Data Communications Network) with parallel processing (within ASP sub-strings), by separating input–output from processing with the ADBs;

(iii) overlapping of sequential processing in the ACUs with parallel processing (in ASP substrings);

(iv) mapping different application data structures to a common string representation within ASP substrings (supporting content-addressing, parallel processing, and a reconfigurable inter-processor communication network).

The numeric-to–symbolic data conversion process (sect. 1) is achieved by first distributing data elements over the identical processors of the ASP substrings and then executing successive tasks on the stored data. Such temporal partitioning of task parallelism ensures that task processing rates are independent of input data rate, owing to data buffering

with ADBs; and scaling of overall parallel processing power simply entails adjusting the number and length of ASP substrings. Indeed, ADB storage and the uniformity of ASP substrings provide much cheaper media for scaling than the expensive processors of a dedicated pipeline.

In summary, the ASP chip architecture offers a highly compact implementation of the SIMD processor discussed above, which exploits the following benefits of associative processing:

(i) Elimination of processor (location) addressing, for the purpose of :

(a) achieving unlimited architectural scalability,

(b) implementing cost-effective fault-tolerance with simple (hierarchical) bypassing of faulty processor substrings, which, until failure occurs, are available for parallel processing,

(c) further minimization of sequential processing overheads.

(ii) Minimization of inter-processor data movement, with high-speed activity transfer between processor subsets and in situ processing.

In particular, the ASP architecture provides an unprecedented opportunity for second-generation MPPs which require dynamic allocation of processors for the flexible and progressive trade-off of parallel processing power for increasing fault-tolerance.

Further descriptions of the ASP architecture can be found in the attached supporting documents (Annexes I and II).

By exploiting the inevitable VLSI-to-ULSI-to–WSI technological trend and state-of-the-art packaging technologies, cost-effective and highly-compact ASP modules can be implemented, which achieve processor packing-densities which are more usually associated with memory components.

In practice, ASP module form factors will depend on the packaging standards adopted for the modular construction systems of particular application environments. Indeed, the availability of such packages is intended to simplify MPP hardware development to the electrical/thermal design of appropriate bus–cooling racks, into which replaceable ASP modules can be plugged, in order to construct a MPP crate compatible with its operational environment.

2.1.1 ASP software

ASP application programs can be written entirely in a familiar block-structured high-level language (e.g. PASCAL, Modula–2, C, or ADA) under a familiar operating system (e.g. Unix, VMS or MS–DOS). Such programs include calls to external precompiled ASP

algorithms and procedures, written and progressively refined (in the same language) by experts, with an intimate knowledge of the ASP architecture, using a set of built-in function and procedure primitives to implement basic ASP operations. Thus, the average ASP application programmer/user does not face the full complexity of parallel algorithm development, but the much less demanding task of selecting and interfacing appropriate code from the hierarchically-organized ASP algorithm and procedure library, and only occasionally resorts to the creation of such code.

Examples of ASP procedures are given in Annexes I and II.

Parallel constructs for ASP programming are being actively investigated in experimental research programmes, but this activity is beyond the scope of this proposal.

The ASP software development system incorporates an ASP simulator, supporting multi-window ASP state and timing displays, to provide a flexible environment for the development of ASP programs.

2.1.2 TRAX-1

Developed by ASPEX–Microsystems, LAA, and the NA35 Collaboration, Trax-I consists of an ASP, with 16,384 Associative Processing Elements (APEs) which has been targeted for the off-line analysis of streamer chamber pictures of very complicated events. As shown in Fig. 8, TRAX-I consists of:

Hardware

- Host and host environment:
  - SUN 3/160
  - Unix and SUN tools

- Intermediate–level control
  - 68020
  - 256 kbytes of program storage
  - 512 kbytes of ASP data storage (RAM)
  - Provision for storage extension
  - VME/VSB interface to host, ASP modules and data capture devices (CCDs)
  - DMA data transfer between host, intermediate-level controller, and ASP modules

- ASP module control
  - Custom built based on a high performance sequencer
  - 256 kbytes of micro-program memory
  - Input scalar buffer with 8 kbytes cache

- Output scaler buffer with 8 kbytes cache
- 16 kbytes scratch pad buffer
- Scalar-vector operations hardware support

• ASP modules
- 4 ASP modules, each comprising 4096 APEs and (2 x 4 k x 32-bits) Array Data Buffer (ADB)
- 64-bit Data Register per APE, supporting bit-serial, byte and 32-bit word data formats
- 5-bit Activity Register per APE
- String connectivity between all ASP modules.

Software

• Host and host environment
- Programming in standard Modula-2, use of Libraries
- Debugging standard SUN-based tools

• Intermediate-level control
- Programming in Modula
- 2, use of Libraries

- Debugging 68020 EPROM monitor

• ASP module control and ASP modules
- Programming Modula-2 based custom language, use of Libraries
- Debugging ASP display windows, performance statistics window, source code trace.

2.2 ASP potential

The progress of ASP hardware and software research heralds the development of a highly-versatile and fault-tolerant line replaceable module (SEM-E) for the construction of modular MPP systems; with significant benefits in reliability, ease of maintenance, and service lifetime achieved with optimal exploitation of WSI microelectronics and packaging technologies.

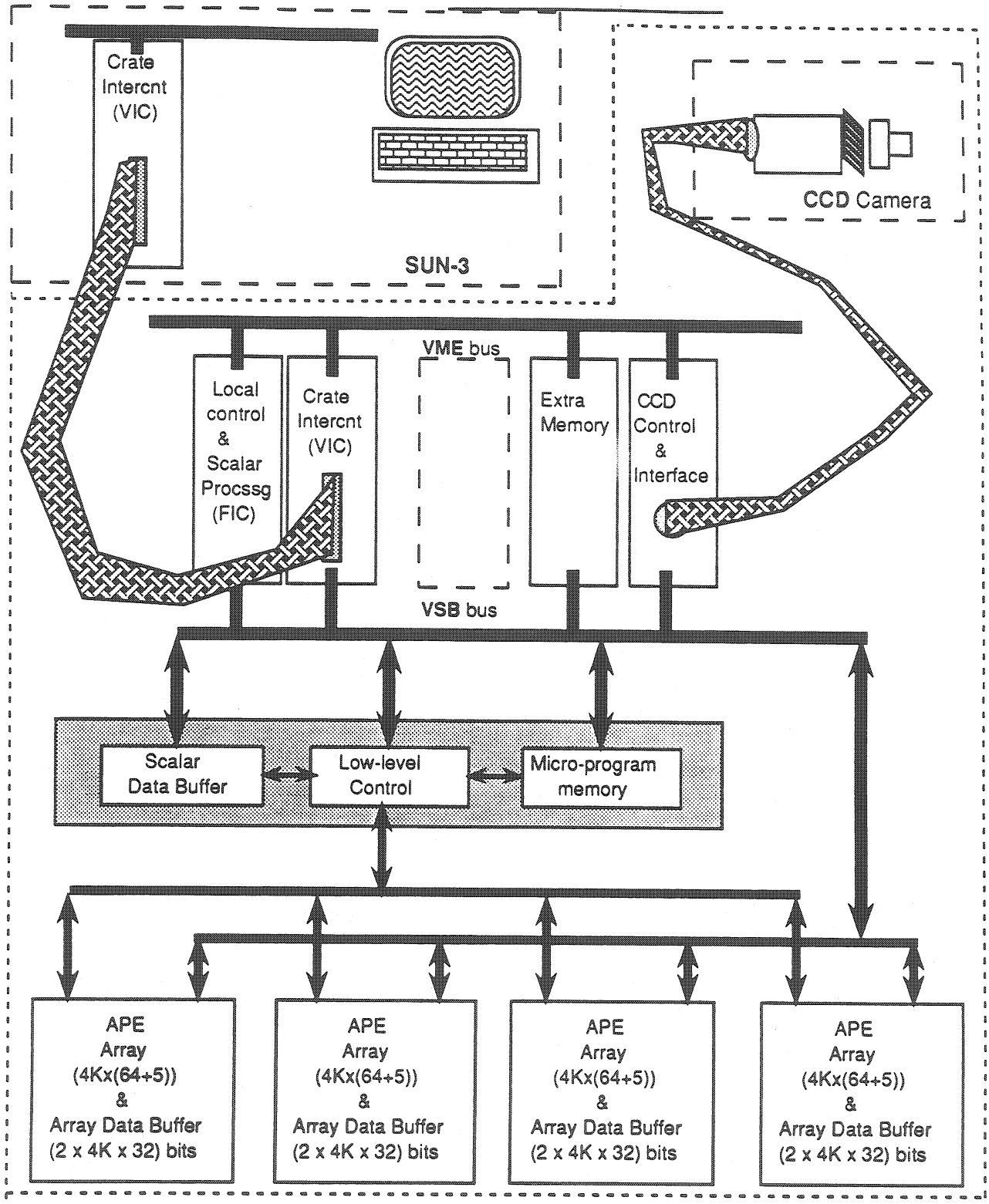


Fig. 8 The TRAX 1 machine. Schematic diagram of the system architecture

In terms of cost-effectiveness, application studies and benchmark evaluations are demonstrating that ASP modules could match and often improve on the performance figures of contemporary MPP implementations; consequently, the competitive edge becomes that of size, weight, power requirement and, especially, cost–performance. Hence, the design targets shown in Table 2 summarize the potential of ASP modules in these terms.

TABLE 2 – ASP module design targets

Configuration	1, 2 or 4 MIMSIMD blocks
Performance	100 GOPS (12-bit adds at 40 MHz)
Input-output bandwidth	320 Mbytes (at 20 MHz)
Number of processors	65,536
Package size	6.4" x 5.88" x 0.6" (SEM-E)
Power dissipation	< 100 W (i.e. > 1 GOPS/W)
Market cost	< $100,000 (i.e. > 1 MOPS/$)

Since they are based on the extrapolation of results gained from experimental VLSI and WSI ASP test chips, the first four ASP module design targets of Table 2 are fairly conservative. Moreover, the cost target includes ample provision for marketing overheads, software support and profit, thereby indicating potential for significant cost reduction for higher volume marketing. In fact, the most speculative target is that of power dissipation, this being a compromise between the 50 W quoted for SEM-Es and the desire to maximize performance. Hence, current research is addressing this issue not only in terms of minimizing power dissipation in WSI circuit design, but also in improvement of the thermal characteristics of appropriate assembly and packaging materials.

Comparison with the 65,536-processor Connection Machine [CM-1: achieving 2.7 GOPS (for 12-bit adds) with a 62.5 Mbytes/s input-output bandwidth, housed in a 50" x 50" cabinet and dissipating 12 kW, for a market cost of $3M] suggests that ASP modules offer some three orders-of-magnitude improvement in cost–performance and size. More significantly, ASP modules offer simple scalability; for example, a peak performance of 1 TOPS (i.e. 1000 GOPS) could be achieved with only 10 modules, within less than a cubic foot, and dissipating 1 kW.

2.2.1 ASP module performance

Extrapolation (from the results of evaluation experiments with 2 μm CMOS VLSI ASP test chips and a 20 MHz clock-rate) enables the forecast of timings for addition and

multiplication (1.2 μm CMOS chip fabrication technology and a 40 MHz clock-rate) presented in Tables 3 and 4.

TABLE 3

Bit-serial arithmetic timings and equivalent performance
(in GOPS, i.e. 10^9 operations per second) assuming a 65,536-APE ASP

	8-bit	16-bit	32-bit
Add/subtract	0.45 μs (146)	0.85 μs (77.1)	1.65 μs (39.7)
Scalar-vector multiply	2.90 μs (22.6)	10.0 μs (6.55)	37.0 μs (1.77)
Vector-vector multiply	5.15 μs (12.7)	18.8 μs (3.49)	71.9 μs (0.911)

TABLE 4

Bit-parallel arithmetic timings and equivalent performance
(in GOPS, i.e. 10^9 operations per second) assuming a 65,536-APE ASP

	8-bit	16-bit	32-bit
Add/subtract	0.15 μs (54.6)	0.15 μs (27.3)	0.15 μs (13.7)
Scalar-vector multiply	0.55 μs (14.9)	1.10 μs (3.73)	3.20 μs (0.640)
Vector-vector multiply	1.80 μs (4.55)	4.95 μs (0.827)	12.4 μs (0.165)

To indicate typical DSP application performance, Tables 5 and 6 forecast digital filtering and fast Fourier Transform (FFT) timings.

TABLE 5

Digital filter timings (bit-serial mode) and equivalent performance
(in GOPS, i.e. 10^9 operations per second) assuming a 65,536-APE ASP and
equal bit-precision for samples and weights and full-precision results

	8-bit	16-bit	32-bit
4-tap	0.15 μs (54.6)	0.15 μs (27.3)	0.15 μs (13.7)
8-tap	0.55 μs (14.9)	1.10 μs (3.73)	3.20 μs (0.640)
12-tap	1.80 μs (14.9)	4.95 μs (0.827	12.4 μs (0.165)

TABLE 6

FFT timings (bit-serial mode) and equivalent performance (in MOPS, i.e. 10^6 operations per second) assuming a 65,536-APE ASP and 8-bit samples and full-precision results

256-points	752 μs (0.340)
512-points	874 μs (0.146)
1024-points	977 μs (0.065)

For higher performance levels, the results could simply be scaled by module replication.

It should be noted that the performance forecasts presented in Tables 3 to 6, are representative of those expected for the long-term target ASP module (see above) and not interim ASP prototypes such as TRAX-1.

2.2.2 Performance benchmark results

Of course, such performance forecasts refer only to the low-level arithmetic operations of the numeric-to-symbolic data conversion process of real-time signal and data processing. Indeed, more realistic performance indication can be gained from application benchmarks, which include all processing from data input to the output of the required response. To this end, the ASP architecture has been evaluated in terms of the 2 US DARPA Image Understanding benchmarks set in 1986 and 1988 and the CERN LAA benchmark (which includes TRAX-1) in 1989. Although beyond the scope of this proposal, the results (see supporting documents: Annexes III, IV, and V) nevertheless indicate major cost–performance advantages for ASP modules compared with contemporary parallel computers.

3. OBJECTIVES

3.1 The Pilot Project: an intelligent on-line camera

Parallel processing and in particular associative parallel processing can solve the computational bottleneck of a large variety of problems in very different fields, some of which are of great interest to this Collaboration. But in order to focus our efforts on the development of the necessary hardware and software platforms a leading project was selected which will demonstrate in an experiment at CERN the high rate on–line data acquisition and analysis of events with at least 1 Mbyte of information each.

The most straightforward detector that can produce the required data rate and which is useful in several detectors for high–energy physics experiments is an image intensified CCD camera with 1 k x 1 k pixels readout at close to video rates. This is the choice of the collaboration as a Pilot Project for MPPC. A sketch of the proposed camera is shown in Fig. 9. Such a system can be used to read out avalanche chamber events in relativistic heavy ion collisions [7] in which hundreds of particles are produced in the final state, or for large–area RICH counters being developed or for the readout of scintillating fibre detectors for future hadron collider experiments [2].

The CCD sensor selected for the camera is the Thomson THX 31156 [8] with 1024 x 1024 square pixels of 19 μm^2 area. It has been chosen because of its 100% sensitive area, its large dynamic range, and good signal–to–noise ratio and quantum efficiency. It will be coupled with fibre optics to a four–stage EMI image intensifier for the detection of events at very low light levels.

The camera head should have two functional modes; one being the TV mode in which the image can be displayed directly on a TV monitor and a digitizing mode in which the image of the triggered event is sent digitized to the ASP modules. The first mode is necessary for the focusing of the camera and the debugging of the detector being imaged. The digitalization of the 1 k x 1 k CCD chip will be done using the four outputs each at 10 MHz on 10 bits, using double correlate sampling for a good signal–to–noise value.

In the acquisition mode a pedestal image of the inherent noise of each pixel has to be subtracted. For this reason a blank picture will be stored in a buffer on command. For real images after pedestal subtraction the 10 bits will be reduced to 8 by using a look–up table. The stream of 8 bit data will be sent to the ASP; the detailed bus and the manner in which it will be done still has to be determined. It will be a great advantage to use this architecture because it allows processing of the image on-line with the data taking, using all the features of the 65,536 ASP including fast data exchange, fault tolerant capability, and reconfigurability for fast intercommunications between APEs.

On the acquired images iconic and symbolic image processing will be done to reconstruct tracks and RICH patterns over the whole image. The full image will be analysed by mapping and processing patches. In order to get the highest efficiency possible, algorithms will be developed taking full advantage of the parallelism and of the ASP architecture.

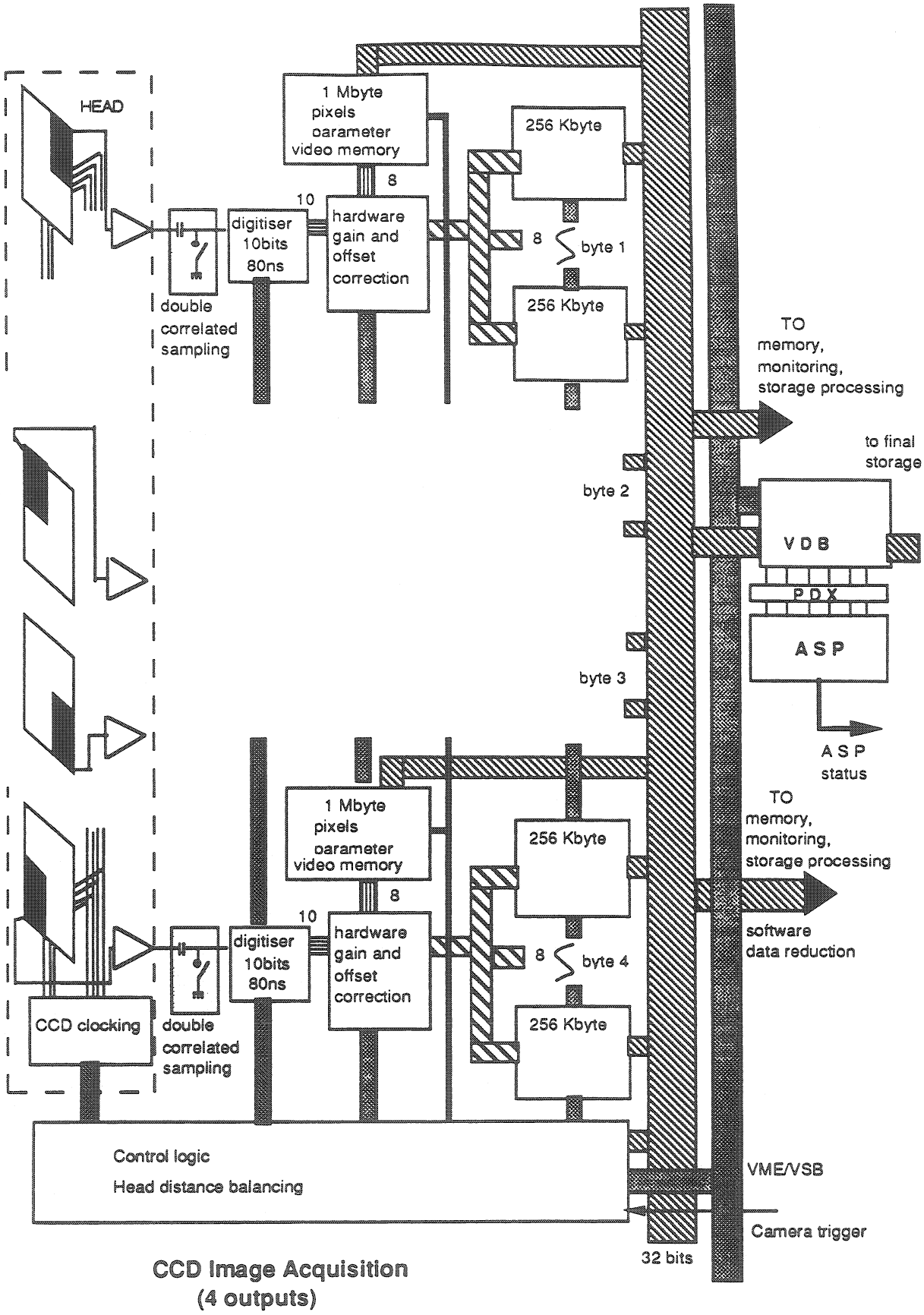


Fig. 9 The Pilot Project : a CCD 1 megapixel on-line camera

3.2 The contribution of massively parallel processing to high–energy physics

There are many ways in which MPPs can have an impact at CERN and on its environment. Among these only the main class of applications will be indicated.

MPPs can be imbedded in future detectors for a very powerful and high throughput front end that can be used as a second – or third – level trigger, in particular in future hadron collider experiments. They can be used for on-line, real time event analysis in all detectors utilizing CCD sensors as readout elements. They can be configured into a very powerful search tool for very large data bases as is needed in most experiments for the analysis of the data summary tapes (DSTs), for example using n-tuples in PAW.

It is possible that neural network methods will be of use to solve many problems in high energy physics because the required computational power is offered by the MPPs.

In the Lattice QCD calculations, MPPs can again bring a great advantage.

3.3 The contribution of massively parallel processing to other sciences

There would be a very large variety of other possible applications for an ASP-based machine equipped with a good software kernel.

First of all, intelligent image processing itself is an outstanding problem in many fields, such as robotics, earth sciences, astronomy, and medicine. Algorithms developed for high–energy physics could be adapted to these other applications, or conversely. This is one of the objectives of the "Laboratoire de Traitement des Signaux" (LTS) at EPFL.

But several other unrelated projects are also in urgent need of a good MPP machine. A non-exhaustive list of such projects at EPFL follows :

(i) Digital TV

The LTS is also involved in a very ambitious program aimed at the optimal compression of digital images. New approaches have been explored since 1980, based on human visual perception models, and allowing compression factors larger than 100 to be reached. The problem is that these methods cannot be used for very important practical applications, such as digital TV, using only the available technology, the amount of computation being much too large for conventional machines to manage in real time. It is hoped that the ASP technology will provide a solution, and make feasible applications such as videophone and high definition television.

(ii) Cellular automata

During the last few years, some people have been trying to demonstrate the efficiency of cellular automata for simulation of various physical phenomena (diffusion, convection,

combustion, etc.). This approach allows a direct representation of the model, to be given avoiding the approximations and computational difficulties of the numerical methods. Cellular automata are also easier to adapt to an increased complexity of the model. A frontier computing capacity such as an ASP based machine would much improve this demonstration and open up a new range of applications.

(iii) Neural networks

There is at EPFL a rather large number of research projects which are in some ways linked to the artificial neural network approach. An international conference on this subject will be held in October at the Department of Electrical Engineering. People working in this field have formed a very active special interest group (CARNAC), trying to identify the best solutions for their common problems. Interest was expressed for the ASP machines, which could be an excellent tool for neural network topology simulations.

4. STRATEGY

4.1 The MPPC project

This project consists of the setting-up of a strong collaboration for the study of the hardware and software development of massively parallel processing based on ASPs and its applications in high–energy physics and for other sciences in view of the present and future need for a drastic increase in computing power.

A precise description of the proposed working programme is to be found in sect. 5 and the necessary required support for the project is estimated in sect. 6.

In the proposed project we can find three important periods:

(i) ***First period: The learning period*** (2 phases of 3 months starting ~ October 1989)

Some of the contributors to this project have no experience in massively parallel processing. They want to enter the field of ASPs, as many of us already did, by following an accelerated training course covering ASP technology, ASP programming, and an introduction to the ASP simulator. Future collaborators, who will join the project for administrative reasons probably not before the beginning of 1990, will also have to follow a full training course in this very new science and consequently a second course must be foreseen for the beginning of next year. After this, all the collaborators will make a further evaluation of their needs and requirements for continuing with the two-year project.

First Milestone: foundation workshop of the MPPC project.

(ii) Second period: The projects (about 18 months)

The MPPC project will actually start with the foundation workshop. During this second period, the software and the hardware development work will be organized and performed according to the decisions taken during the major foundation workshop. The first work will be to produce a detailed project schedule chart including all possible tasks from the beginning of the design to the final acceptance test. It consists, for the hardware part, of the full detailed specification and design of the 65,536 ASP modules and all the relevant necessary boards and crates. It also includes the choice of host computer and bus interconnection. The specification and design of the Pilot Project will be done in parallel, including the acquisition and digitization electronics, the full grey image storage, and the vector data buffer (hybrid version first) for the data exchange with the ASPs. The development of the software will be performed during the same period for the two proposed applications: image processing of RICH counters and streamer chambers. The allocation of the responsibilities will be done within the Collaboration for the construction, the assembly, and the testing of the hardware parts as well as for the development of the software for the data acquisition and for the applications with on-line processing. Acceptance tests and benchmarks will be precisely defined. This will be the:

Second Milestone: acceptance of the hardware and software of the on-line camera.

(iii) Third period: Demonstration and Evaluation (about 6 months)

The choice of the location of the on-line camera for the demonstration and evaluation of the performance will depend on the existing experiments at CERN by the end of 1991. The choice will be based on two important criteria: the implementation feasibility of the equipment in the experiment and the physical output which could be expected. The work will consist of three phases: installation in the experiment, running and data taking, data analysis with a comparison between on-line and off-line results.

Third Milestone: MPPC Open Review.

This will be the final milestone before the application of this new technique in future experiments: presentation of the final results ofc the MPPC projects.

4.2 The MPP software platform at EPFL

Several groups at EPFL and elsewhere are facing enormous computation power problems, which motivated purchasing a Cray/1 in 1986 and a Cray/2 in 1988. This latter machine is already saturated, while many big projects have not yet started to run That is one of the reasons why the EPFL management board recently recognized the need to promote research work on massively parallel processing. It was decided to launch a two years start–up program not only to explore the numerous approaches, but also to initiate a change of

behaviour among computer professionals in order to fill the gap which has existed until now between machine designers and end-users : in the realm of parallelism, programmers of applications cannot any longer ignore the hardware architecture, nor can the designer ignore the specificity of the applications to be run on the machine. The goals and time scale of the MPPC project exactly fit the profile of this programme. It is hoped that the activity generated by the setting of a software platform at EPFL will contribute to achieving its objectives.

ASP programming is not trivial and cannot be learned quickly : it needs at least three months of training before one is able to produce valuable code. EPFL is expected to take care of this training for the whole Collaboration with the assistance of ASPEX. Establishing a sofware platform is therefore necessary to allow application programmers to develop specific algorithms or debug their programs. Experienced or trained people from the whole Collaboration will contribute to creating a complete system to control ASP modules, producing a library of basic procedures and other tools, and shaping a user-friendly parallel computing environment adapted to the MPPC range of applications.

The whole software structure must be carefully designed from the beginning, following a hierarchical structure more or less imposed by the hardware architecture. Each project will define its needs in a tree-like way, keeping in mind that the basic elements should be as general purpose as possible. This will allow for identifying the common needs, and defining basic "primitives", i.e. reusable "sofware components" as well as more specific ones. The list of wishes will be partially satisfied with existing material, possibly modified, and the missing components will be developed by the Collaboration. All the software elements will then be assembled at EPFL in a layered structure, starting from the most basic ones ending up with those most specific to the applications (bottom–up strategy).

4.3 The MPP hardware platform at Saclay

At Saclay, it is proposed to set up a hardware platform for the development of the boards which have to be constructed for the MPP project. This implies a close collaboration between ASPEX and Saclay. Saclay objectives are to participate in the development of ASP technology in order to establish a knowledge–base and acquire an experimental capability in on–line massively parallel processing. The collaboration is intended to benefit Physicists and to support the hardware and software aspects of MPPs which may be incorporated in front–end instrumentation for the next generation of accelerators (e.g. LHC and SSC).

4.3.1 List of activities

This consists of R & D on the following components:

(1) ASP chip with 64 APEs (which already exists).

(2) ASP hybrid module with 4x4 ASP chips.

(3) ASP board:

(3.1) Test/evaluation board with 1 ASP hybrid module, including 2 levels of inter-processor communication (short and long range and overlapping of chips for fast link between APEs), buffer memory, and one I/O channel.

(3.2) ASP board with at least 3 x 3 hybrid modules including appropriate buffer memory and multiple I/O channels.

(3.3) Full board with 16 ASP hybrid modules (16384 APEs), buffer memory and multiple I/O channels the number of which is to be specified.

(4) Duplication of full ASP board for ASPEX.

(5) System crate and host computer.

(6) Integration of the MPP with one ASP board.

(7) Duplication of ASP boards and system crate for CERN.

(8) Upgrading of the CERN and ASPEX sites to a 64 K laboratory MPP prototype.

(9) Upgrading of the laboratory MPP prototype to the final MPP prototype for reproduction for end-users (possible candidates being the collaboration partners and others).

4.3.2 Responsibilities

Item	Responsible for the work
(1), (2)	ASPEX
(3), (4), (5)	Saclay and ASPEX
(6), (7), (8), (9)	Saclay

4.3.3 Construction subcontractors

It is agreed that:

(i) Saclay

- will make its CAD system for board design available free to the Collaboration;
- will make its manpower effort available free to the Collaboration;
- will need to be reimbursed at their purchase cost for the expenses of external components used on the boards built for ASPEX and CERN.

(ii) ASPEX

- will contribute half of its effort in manpower (half of the salaries);
- will contribute half of the cost of the hardware components;
- the other half (manpower and components) needs to be provided by the Collaboration.

4.4 CERN involvement

Apart from the fact that CERN has been taking the initiative for trying to set up this project, it should be strongly involved in MPPC as one of the major collaborators for the evaluation of this new technology of the future in data acquisition and computing. Therefore the coordination of the project will be done by CERN under the responsibility of a MPPC-CERN Project Leader. The Pilot Project of the Collaboration will be assembled and tested at CERN. The demonstration and the feasibility study of this Pilot Project will be done in a physics experiment at CERN for the real–time data–acquisition and on-line data analysis. Consequently the involvement of CERN in the MPPC Project is essential and the support in manpower and financing should be given accordingly.

The Collaboration wants to present an ESPRIT proposal and will have the help of CERN for this purpose.

4.4.1 General support

The expected necessary involvement is shown by an estimate of the number of people required for each task. The details of the structure and task force will be worked out later by CERN in the case of approval of the MPPC project. The motivation for studying the massively parallel processing, already stressed in the Introduction, is so strong that a major effort is needed. Because of the strength and value of the proposed Collaboration, it is worth starting with a well–structured group in CERN for the MPPC which will have to grow according to the working programme.

The activity of CERN will consist of:

(i) ***Coordination of the project:*** 1 person and secretarial support

- Planning, training, workshops, administration (personnel and budget), ESPRIT proposal, progress reports, safety, etc.

(ii) ***CCD platform:*** 5 persons (see details in sect. 4.4.2)

- Designing, hardware coordination, testing of boards, control, monitoring, laboratory support, CCD interface, power supplies and environment tools, etc.

(iii) ***Mechanics and optics:*** 1 person and laboratory support

- Camera head, lens, image intensifier with all its equipment, crates, test bench, installation, survey, etc.

(iv) ***Software:*** 2 persons

- Development of algorithms, data acquisition, system programming, data analysis ASP procedure library, host computing, system integration, acceptance tests, etc.

(v) *Physics:* 1 person + the host collaboration

- Preparation for the demonstration and evaluation, understanding of the detectors, application software specification, running on the CERN experiment, evaluation of the physics results, prospects, etc.

4.4.2 CCD hardware and software platform

CERN will be responsible for the complete image acquisition system based on a full–frame CCD image sensor, 1024 x 1024 pixels, manufactured by Thomson as described in sect. 3.1. Consequently, it is proposed to establish a platform at CERN for this task. The work will consist of the following R & D programme:

(a) A first prototype will be designed and constructed in close contact with Thomson. The architecture will be tested and optimized without any risk by using a cheap CCD sensor. The basis of the architecture is described below.

- A camera head with the following components:
  - CCD clocking and control,
  - Four amplifier outputs,
  - Video output for direct monitoring,
  - Double correlated sampling,
  - Line drivers of the analog signal to the control card,
  - Line drivers for control signals.
- A control card based on a VME or Fast Bus standard with :
  - 10 bit digitizers (one for each of the four outputs of the CCD),
  - One Mbyte memory bank for the blank storage,
  - Hardware substractors and rounder for full significant 8 bits data,
  - One bank (at least) of one Mbyte memory for local buffering,
  - First simple compaction for storage,
  - Interface logic with the bus.

(b) A second monitoring facility will allow the visualization of the contents of a memory bank. It will be installed on the card itself or if there is not enough place on a brother card without connection to the bus. The direct connection to a mass storage system is essential for off-line comparison and analysis.

A close collaboration with ASPEX, Saclay and Thomson is necessary to ensure perfect integration in the general system crate (bus and protocol).

(c) A duplication of the system crate, a test bench, and all the necessary facilities must be installed at CERN.

After the construction and tests a duplication of all the hardware will permit safe installation of the final CCD while carrying out improvements.

(d) Some specific software must be developed in close connection with the camera–designing team for the direct control of the camera, acquisition and storage of the blank picture, first compaction and tests.

Together with one CERN staff member responsible for this hardware and software CCD platform, the required support and contributions are:
- an engineer or a senior technician with experience in computing, buses, hardware design,
- two technicians for other designs and contructions,
- a system software support with knowledge of the assembler language used.

The external components will have to be funded by the Collaboration.

Thomson, as described in sect. 4.5.4, will supply the sensors and provide full support in the design of the camera.

4.5 Motivations, applications, and involvement of the other collaborators

4.5.1 ASPEX-Microsystems

ASPEX strongly supports the MPPC project and intends to contribute major effort comprising four engineers for software and hardware development and appropriate management. In the proposed Collaboration, ASPEX will be responsible for commissioning, testing and delivery of the ASP chips. ASPEX will also collaborate with Saclay in the design, testing, evaluation and integration of the MPP hardware platform. In addition, ASPEX will work closely with EPFL to provide education and training of the collaborating groups and it will assist EPFL to establish the collaboration software platform. The Managing Director, Professor R.M. Lea, will also assist the CERN Project Leader in the coordination of the project. Lastly, ASPEX will endeavour to provide the collaboration with ASP components from the most advanced technology available for MPP prototype development.

ASPEX in general, and in particular within the context of the MPPC, does not regard CERN as a customer for ASP commercial products, but as a suitable site for the demonstration of the potential of ASP technology. Nevertheless, ASPEX is a young and growing company and, as such, lacks the financial resources to fund this major effort. For this reason, and in view of its enthusiasm to participate in the MPPC, ASPEX is prepared to follow the established procedures of widely accepted European programmes (e.g. ESPRIT).

Accordingly, ASPEX would require half of its total costs, including manpower, to be supported by the Collaboration. It should be noted that ASPEX intends to include no profit in its costing of the MPPC project. Moreover, it is important to note that ASPEX, within this collaboration, is not seeking financial support for ASP technology development.

Finally, an important condition of its participation is that, as the only commercial partner of the collaboration, ASPEX must have exclusive rights to the commercial exploitation of the developed MPP hardware and software.

4.5.2 Brunel University

Independently of ASPEX, Brunel University hosts many student ASP related application software projects. Accordingly, Brunel would welcome the opportunity to join the MPPC and could make a significant contribution, in conjunction with EPFL, to the development of the software platform (sect. 4.2). However, this contribution would require the appointment of a Research Associate at Brunel. Brunel itself has no funding mechanism to support this post and therefore the cost would have to be borne by the MPPC.

4.5.3 Geneva University

A group of physicists from the University of Geneva is working at CERN for an experiment which should be installed in the H1 beam using a large–aperture tracking facility: the CERN Omega Spectrometer.

The aim of the experiment is the measurement of the hadronization products of coloured objects such as diquarks or triquarks produced in a hadronic collision where a direct photon is emitted. This is a continuation of the WA70 experiment (Hadronization of Fragments in Prompt Photon Events), where direct photon production cross sections were measured. This includes the identification of charged hadrons by a RICH detector now under development, for use at the Omega Spectrometer.

The **present status** of the project is the following: light emitting chambers sensitive to Cherenkov ultraviolet (UV) photons have been built following a prototype developed by Charpak and collaborators, and are now under test. This includes the readout by a CCD camera. Single UV photons have been detected with a sufficient sensitivity. At present, the team consists of CERN and University of Geneva members but it is necessary to enlarge the Collaboration. A Letter of Intent [9] has been sent to CERN, but no proposal as yet.

The **RICH** consists of an already existing radiator and a 10 m radius mirror, to be equipped with three light chambers 1.2 x 1.6 m^2 each. The readout is foreseen as six frame transfer CCD cameras, 2 x (288 x 385) pixels each. The data–tracking rate is 100 events per 1.6 s SPS burst, every 14 s, selected by a fast trigger for large transverse momentum events, from about 2 x 10^6 interactions per burst.

The **readout** of the cameras is foreseen to be done in 9.6 ms with a 6-bit intensity per pixel, the storage section of the CCD being used as an intermediate buffer. Data compression is made by hardware processors and the resulting information is stored through VSB in FIC

8230 processors during the burst. The data are accessed between the bursts via VME (or VSB) for recording and monitoring on-line pattern recognition.

The **data reconstruction** is done on-line on a sample of events and off-line. The RICH pattern recognition off-line program can be done in a conventional way. Although no funding is yet available for the development of on-line processing, the need for the fast image recognition in this experiment makes the use of parallel processors very desirable.

4.5.4 GSI

The Gesellschaft für Schwerionenforschung (GSI) participates in two experiments, NA35 and WA80, with ultrarelativistic heavy ions at CERN. Utilizing beams of 16O and 32S at 200 GeV per nucleon these experiments study the behaviour of nuclear matter at extreme conditions of density and temperature. In particular, they search for the transition of hadronic matter to a plasma of quarks and gluons that has been predicted by Lattice QCD calculations.

On central collisions of the S beam with heavy targets more than 400 charged particles and as many neutrals are produced in the final state and a large fraction of them are detected by the present experiments. Effectively the multiplicity of detected particles and the complexity of the events is larger than that of the events that will be produced at the Large Hadron Collider (LHC).

The analysis of the events, in particular for the NA35 experiment, where the trajectories of the charged particles are recorded by a streamer chamber with three stereo views on 70 mm film, will require a tremendous computing power that exceeds by orders of magnitude that presently available from commercial systems.

The WA80 Collaboration has solved this problem by producing the data summary tapes on an ACP farm of processors that, in particular, do the analysis of the fine–grained electromagnetic calorimeter Saphir. In NA35 the streamer chamber data is digitized in 6 Mbytes per view and analysed in a factory of operator controlled digital measuring workstations. This solution allows only the analysis of a small fraction of the events, since it takes more than 14 hours to fully reconstruct an event.

In order to increase the throughput of analysed events an automatic data analysis chain has been developed (Fig. 10), based on image processing and image understanding techniques (Fig. 11). For such a scheme to be practical it is necessary to have a machine with processing power of the order of 20 GOPS on 8 bit words and with a very large flexibility to follow the iconic to symbolic processing nature of the analysis.

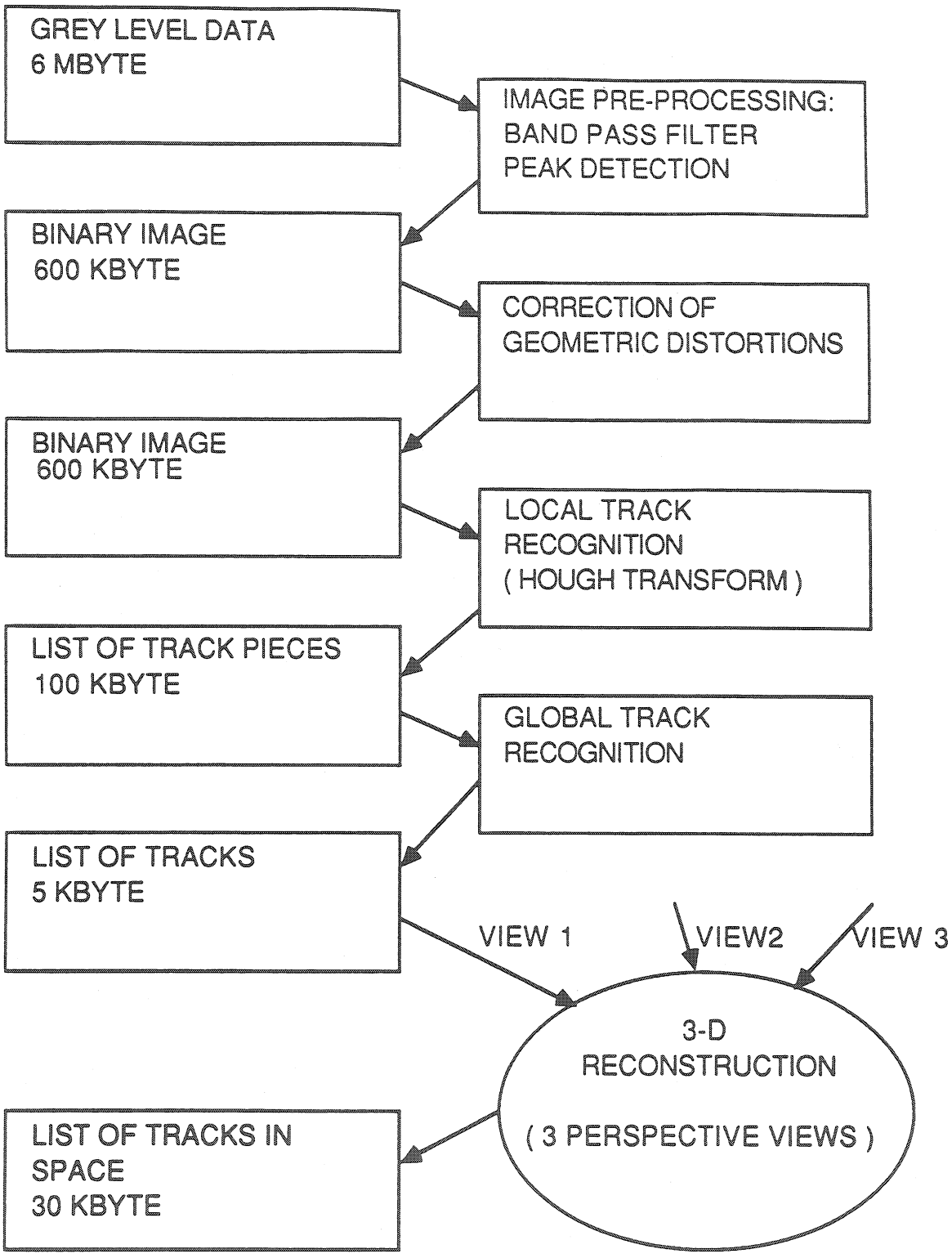


Fig. 10 - NA35 image processing software

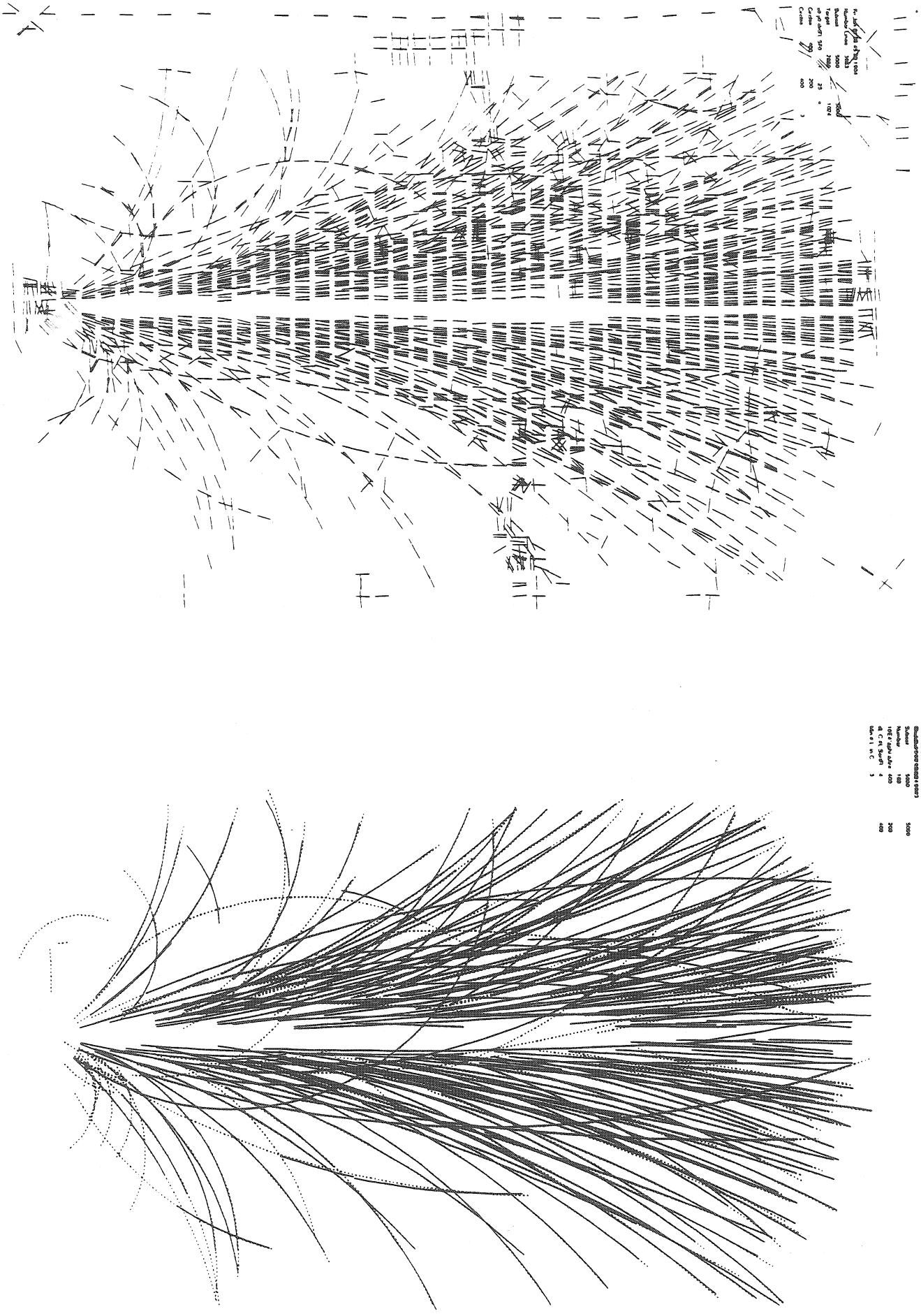


Fig. 11 NA35 image processing of a typical event

These kinds of requirements can at present only be met by second–generation massive parallel processors. Therefore under the leadership of A. Sandoval and in collaboration with ASPEX Microsystems and LAA, the NA35 Collaboration is developing TRAX-I, an Associative Parallel Processor with 16384 processing elements.

The developments within the NA35 TRAX–I project in terms of algorithms and software developments will be available to the MPPC Collaboration. The development of an intelligent camera readout for the Thomson THX–31156 CCD with 1K x 1K pixels being done at present by NA35 can be used as a basis for the MPPC Pilot Project.

The motivation of GSI in the MPPC is to explore the use of the ASP for on–line data acquisition and analysis in view of its future projected experiments. NA35 and WA80 are developing large–area multistep avalanche chambers as multiplicity detectors for charged particles and for the readout of RICH detectors for particle identification. These detectors would greatly benefit from the developments of the Pilot Project.

Still further in the future, GSI plans to participate together with other institutions in a new series of experiments utilizing the proposed lead beams at CERN [10]. For these experiments over 1000 particles per event are expected to be measured at high rate. Very large volume tracking chamber RICH counters and fine–grained calorimeters will be employed, producing hundreds of Mbytes of digitized data per event which has to be processed and stored. For this, very powerful online processors are required for data reduction and analysis. Our aim is to utilize embedded ASP modules developed in this Collaboration which will provide the required hundreds to thousands of GOPS.

The detailed participation of GSI in terms of manpower and money has to be defined. A request is being presented to the GSI authorities to this effect.

4.5.5 Thomson

Thomson-CSF has been involved in work concerned with CCDs activities since the time of their invention in the early 1970's. During the following years this activity has grown rapidly in size and expertize to become a leading one in this field. The main strength of TMS (Thomson Composants Militaires et Spatiaux) consists in its constant effort in following, thanks to its extensive know-how in special sensor configurations, the requirements of the most challenging professional markets in space, military, scientific and, medical applications.

TMS offers, through this cooperation, to provide full support in the design of the camera based on the 1024 x 1024 full frame CCD image sensor (THX 31156), which has been chosen for its characteristics in terms of size, dynamic range, and signal–to–noise ratio. Provided that the use of thinned backside illuminated devices should be required, TMS is in

a position to propose an extension of the work carried out at the present time on smaller resolution image sensors (512 x 512).

Furthermore, the design of a specific high–rate CCD could be proposed on the basis of TMS's customs device facilities.

In any case TMS will offer full support in the definition of the product (architecture, specification. :c.), including operating conditions (drive boards, etc.), and the following of the project (progress meetings).

5. PROGRAMME OF THE WORK

A detailed list of activities and a provisional schedule for the MPPC has been established. This constitutes the flow chart of the proposed project.

5.1 Activities

1. ASP technology training
   - 1.1 Initial workshop on ASP architecture/applications and software
   - 1.2 ASP course
2. MPPC foundation and project definition workshop
   - 2.1 ASP configuration
   - 2.2 ASP system software and programming environment
   - 2.3 Pilot Project and demonstration criteria
     - 2.3.1. Optics (image intensifier and lenses)
     - 2.3.2 CCD camera head
     - 2.3.3 Digitizer
     - 2.3.4 Host computer
     - 2.3.5 Data acquisition and analysis software
   - 2.4. Other project definitions and demonstration criteria
     - 2.4.1 Application software
3. System specifications
   - 3.1 Provisional specifications
   - 3.2 Interim specification workshop
   - 3.3 Revision of specifications and definition of test criteria
   - 3.4 Final specification workshop
   - 3.5 Validation of the final specifications
4. MPP and CCD prototype design
   - 4.1 Sub-system design
   - 4.2 Establishment of MPP software development platform at EPFL
   - 4.3 Establishment of MPP hardware development platform at Saclay

4.4 Establishment of CCD hardware development platform at CERN
4.5 Design verification
4.6 Design reviews
4.7 Performance estimate

5. MPP and CCD prototype construction
5.1 Component procurement
5.2 Sub-system assembly
5.3 System integration and testing

6. MPP and CCD prototype laboratory demonstration
6.1 Pilot MPP prototype
6.1.1 Software commissioning
6.1.2 Demonstration
6.2. Non-Pilot MPP prototypes
6.2.1 Software commissioning
6.2.2 Demonstrations

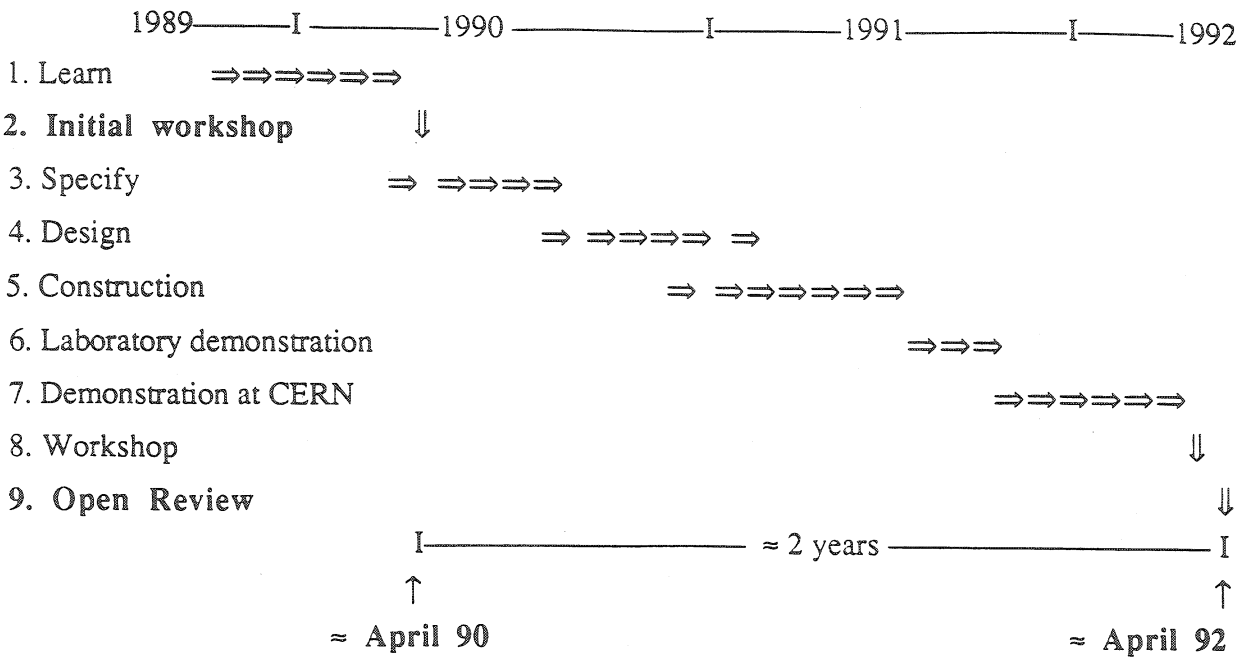
7. On-line MPP-CCD prototype demonstration at CERN
7.1 Selection of the host physics experiment
7.2 Set-up of the Pilot Project demonstrator
7.3 Commissioning of demonstrator software
7.4. Data taking
7.5. On-line and off-line analysis and evaluation

8. MPPC project progress workshops
8.1. Pilot Project
8.1.1 Relevance to physics goals
8.1.2 Selection of physics experiments for MPP–CCD application
8.2. Other application project proposals

9. MPPC technical achievements; Open Review
9.1 Presentation of MPPC results
9.2 ASP technology; state-of-the-art
9.3 MPPC prospects
9.3.1 Impact of MPP technology on high-energy physics
9.3.2 Dissemination of knowledge of MPP technology within the physics research community
9.3.3 Application of ASP distributed processing power in physics detectors

10. Replication of the final MPP and CCD prototypes for end-user applications
11. Installation and support of final MPP and CCD prototypes in end-user applications
12. Participation in on-line high-energy physics experiments based on final MPP and CCD prototypes

5.2 Time scale



The MPPC Programme can be summarized and presented according to major phases with corresponding important milestones which allow to show the necessary cash flow to be shown:

- **Phase 0: Learning & MPPC Foundation**

  Activities 1 and 2 of the list given in sect. 5.1

  Tasks:

  • Establish common foundation platform for the effective support of the MPPC.

  • Define programmes for the MPPC including the Pilot Project (on-line CCD camera).

  Tools:

  • Learning of the existing ASPs technology, software and hardware:

  - Training at EPFL: course, simulator (SUN host), TRAX1 machine.
  - MPPC Foundation Workshop.

  ≈ 6 months: two periods for learning (a training workshop followed by a full course) and the organization of the major foundation workshop. Budget based on ASPEX the latest quotations sent to EPFL.

- **Phase 1: Specification and design**

  Activities 3. and 4. of the list given in sect. 5.1

  Tasks:

  • Specification and high level design of the 64 k ASPs hardware and software boards

- Specification and high level design of the CCD Pilot Project
- Certification of the designs.

≈ 9 months with two Specification Workshops and a Design Review

- **Phase 2: Construction and laboratory tests**

Activities 5. and 6. of the list given in sect. 5.1

Tasks:

- Component procurement, system assembly, and the testing of software/ hardware prototypes testing.
- Certification and laboratory demonstration of the ASP boards and on-line CCD camera.

≈ 9 months, during which the major financial involvement will begin.

- **Phase 3: Demonstration in a physics experiment**

Activities 7, 8 and 9 of the list given in sect. 5.1

Tasks:

- Implementation of the camera on a real experiment.
- Data acquisition, analysis and evaluation of the results.
- Evaluation of the results from the Pilot and Non-Pilot Projects.

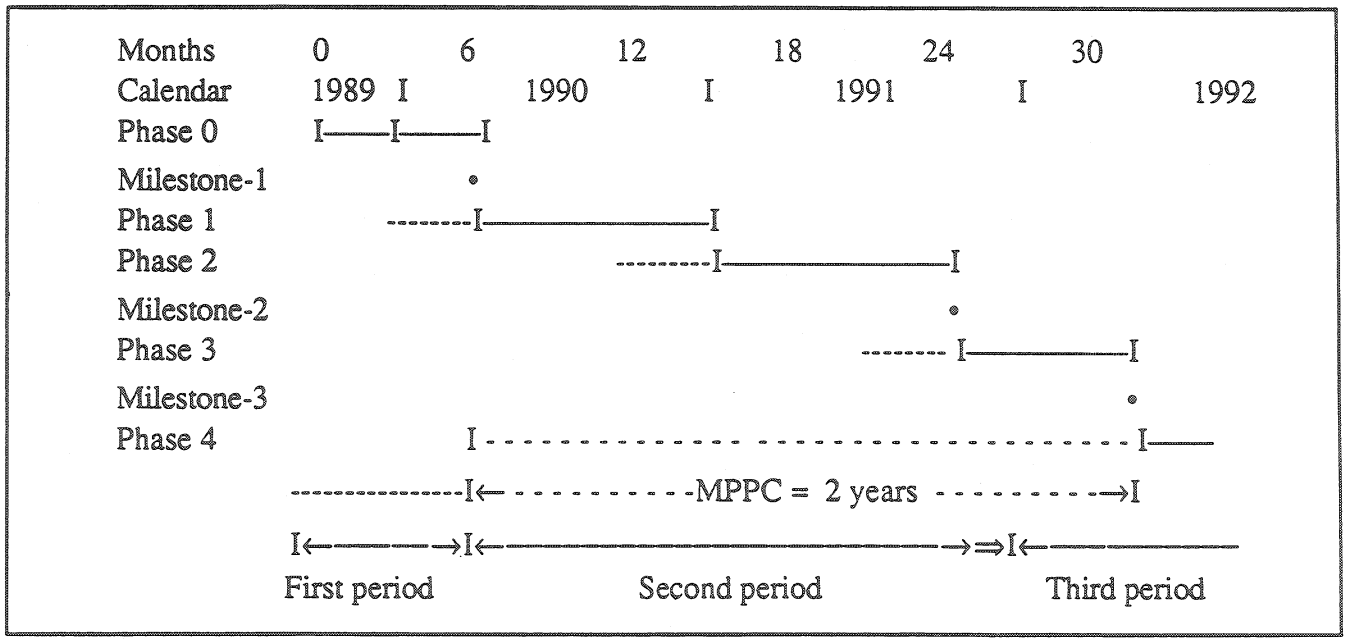
≈ 6 months including the Preparation Workshop and final Open Review.

- **Phase 4: Prospects**

Task:

- Take advantage of the MPPC for launching other projects by all or part of the Collaboration. End-user applications.

According to periods (sect. 4.1) and phases the time scale can be redrawn as follows:



6. RESOURCES

6.1 MPPC personnel

Institution	Responsible
ASPEX	Prof. R. M. Lea
Brunel University	Prof. R. M. Lea
CERN	Dr. F. Rohrbach/EF
EPFL	Prof. M. Kunt / LTS
GSI	Dr. A. Sandoval/NA35
Geneva University	Prof. M. M. Martin / DPNC
Saclay	Dr. P. Borgeaud / DPHPE-SEIPE
Thomson-CMS	Mr. J.-P. Lamarcq

Platforms

Software	EPFL	Dr. J.- J. Dumont /SIC
Hardware	CEA-CEN Saclay	Mr. J.- C. Brisson/DPHPE-SEIPE
CCD	CERN	Mr. J. Feyt /EF

Manpower (Not including the contributors to MPPC and subject to further discussions within each institute)

ASPEX	4
Brunel University	1
CERN	8 including 4 for the CCD platform
EPFL	10 including 6 students
GSI	2 (for third period only)
Geneva University	1
CEA–CEN Saclay	2
Thomson-CMS	1
Total	29

With the contributors the total number of scientists will be 45.

Major sharing of the responsibilities and work

Common trunk (MPP)	Hardware Software	Saclay/ASPEX EPFL/ASPEX
Pilot application	Hardware Software	CERN/Thomson EPFL/CERN / GSI
Non High–Energy Physics Applications	Software	EPFL/Brunel
Other High–Energy Physics Applications	Hardware Software	CERN/Saclay/GSI CERN/Saclay/GSI

6.2 MPPC budget

1.	MPP (common trunk):	Nine ASP-16 k array cards(*) including all control and memory		960 kSF
2.	Pilot Project:	EMI image intensifier/WA44	50 kSF	
		CCD	25 kSF	
		Fast 10-bit ADC converters (4x)	6 kSF	
		Video	5 kSF	
		Fast buffer memory	10 kSF	
		Crate	15 kSF	
		Interconnections	50 kSF	
		Host with mass storage	50 kSF	
		TOTAL	211 kSF	211 kSF
3. Sofware development manpower support				150 kSF
4. Overheads		≈ 20% (of points 6.2.2 and 4/9 of 6.2.1)		128 kSF
5. Unforeseen		≈ 15% (on material only)		86 kSF
6. Simulator and training				100 kSF
		GRAND TOTAL		1635 kSF

The funds should be financed over 3 years (1990–1992) by the Collaboration according to the following cash flow diagram.

1989	10 kSF	only minor overheads
1990	500 kSF	(including 100 kSF for EPFL training)
1991	925 kSF	major involvement
1992	200 kSF	
Total	1635 kSF	

It is very important to point out that the major financial involvement for CERN will occur only after the laboratory testing of the first ASP board developed at Saclay with ASPEX. Consequently, the risk is minimized.

(*) The nine ASP boards are distributed as follows: one board for Saclay as prototype development, four boards for ASPEX as software development tool and four boards for CERN for the Pilot Project.

REFERENCES

[1] A. Zichichi et al., The LAA project, original proposal, CERN, 15 December 1986.

[2] A. Zichichi et al., The LAA project, CERN, status report, CERN–LAA/88-2, 9 September 1988.

[3] ASP: a cost effective parallel microcomputer, IEEE Micro, p. 10-29, October 1988, see Appendix II.

[4] F. Rohrbach et al., The MPPC project, MPPC/89-1, CERN, p.109-111, 30 June 1989.

[5] Thomson CMS, CCD Technical Seminar, London, 7 June 1989.

[6] D. R. Ward, Physics at $\sqrt{s}$ = 900 GeV from the UA5 experiment at the SppS collider, Physics in collision, p. 65-83, eds B. Aubert and L. Montanet, Editions Frontières, Gif-sur-Yvette (1985).

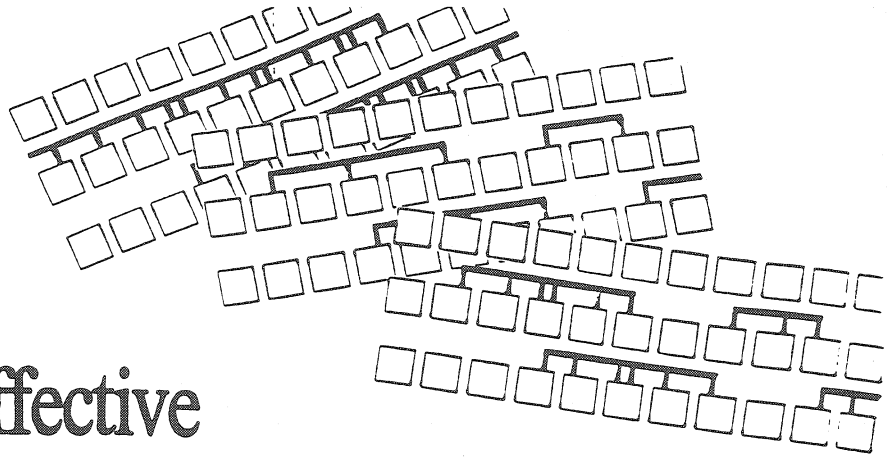
[7] F. Pühlhofer et. al. , Nucl. Instrum. Methods A263 (1987) 366.

[8] G. Beal et al., Opt. Eng. 26, 902 (1987).

[9] University of Geneva–CERN Collaboration, Letter of Intent, Measurement of fragments hadronization in prompt photon events, CERN/SPSC 88–2/I167, 8 January 1988.

[10] NA35 and WA80 Collaborations, Letter of Intent, Large acceptance hadron and photon detector for an investigation of pB–induced reactions at the CERN SPS, CERN/SPSC 89–37/I173 (4 July 1989).

The five Annexes are available upon request at the CERN/EF typing pool.



ASP: A Cost-effective Parallel Microcomputer

Highly versatile, this densely packed parallel processor exploits advanced micro-electronic trends.

R. M. Lea
Aspex Microsystems Ltd.
Brunel University

Associative String Processor microcomputers provide highly versatile components for the low-cost implementation of high-performance information processing systems. By mapping application data structures to a string representation and supporting content addressing and parallel processing, ASP achieves both application flexibility and a step-function improvement in cost-performance figures. This improvement occurs without the loss of computational efficiency usually suffered by general-purpose parallel processors.

The ASP architecture offers cost-effective support of a particularly wide range of both numerical and nonnumerical computing applications while exploiting state-of-the-art microelectronic technology. This technology achieves processor packing densities that are more usually associated with memory components. In fact, we designed ASP to benefit from the inevitable VLSI-to-ULSI-to-WSI (very large, ultra large, and wafer-scale integration) technological trend, with a fully integrated, simply scalable, and defect/fault-tolerant processor interconnection strategy.

Here, I discuss the architectural philosophy, structural organization, operational principles, and VLSI/ULSI/WSI implementation of ASP and indicate its cost-performance potential. ASP microcomputers have the potential to achieve cost-performance targets in the range of 100 to 1,000 MOPS/$1,000 (million operations per second). This gives ASPs an advantage of two to three orders of magnitude over current parallel computer architectures.

The ASP architecture is based on a fully programmable and reconfigurable, homogeneous computational structure emerging from research at Brunel University and being developed by Aspex Microsystems. ASP[1-4] offers particularly flexible (see the box on "Associative Processing") support for structured data processing as indicated by the examples in Table 1.

The breadth of this application range indicates a large potential market for ASP within the aerospace, telecommunications, automobile, and manufacturing industries as well as the commercial, defense, and research sectors. It also demonstrates the importance of application flexibility and architectural extensibility (scaling processing power to match application requirements) as ASP design requirements.

While ASP exploits the opportunities presented by the latest advances in the VLSI-to-ULSI-to-WSI technological trend, it also makes use of the continually improving high-density system assembly techniques (multichip,

0272-1732/88/1000-0010$01.00 © 1988 IEEE

Associative Processing

Many information processing applications require users to reference a set of data elements, associated with a common key, by the value of an associated key (rather than by their physical locations within some storage structure). Examples of data elements might include the selection of those Rover cars sold after 1987 with air-conditioning or those graduate software engineers with two years of experience in Ada programming or those pixels corresponding to a particular intensity value in a computer vision system or those facts and rules that are related to a particular query in a knowledge-based artificial intelligence system.

With traditional von Neumann computers, such data access requires repeated (sequential) navigation through some tree-structured (possibly complex) indirect-addressing mechanism to unique storage locations (where the sought data may or may not exist). This access method results in loss of accessing efficiency and much redundant processing.

In contrast, with associative processing users access the set of data elements in parallel by content addressing and simple association linking. The addresses of such data have no logical significance and only relevant data can be accessed. Moreover, associative processing avoids the additional overheads of sequentially transferring data to an external processor by (parallel) in-situ processing.

Associative processing involves a particularly flexible and naturally parallel form of symbolic representation and manipulation of structured data (sets, arrays, tables, trees, and graphs) processing. Potential benefits include simplicity of expression, storage capacity, and speed of execution over a wide variety of nonnumerical and numerical information processing applications.

Table 1.
ASP information processing applications.

Special-purpose applications

Nonnumerical information processing
- Text processing, database management, office systems, information management
- Information (document) retrieval for information, legal and patents services
- Intelligent knowledge-based, or expert, systems (medical, automotive systems)

Numerical information processing
- Digital signal processing in aerospace, military, telecommunications systems
- Speech recognition in military, business, automotive systems
- Image-related processing
  - Computerized tomography for medical, industrial, geophysical image reconstruction
  - Image clarification, scene analysis, pattern recognition in support of remote sensing (for satellites and surveillance), artificial vision (for robotics, automation)
  - Computerized image generation for graphic arts and special television effects and CAD/CAM (3D image generation, associated database management)

General-purpose applications

- Vector processing for research modeling and design simulation
- Symbolic processing for compilation, translation, theorem proving
- Artificial intelligence processing for fifth-generation (declarative) support of programming languages such as functional (LISP) and logic-based (Prolog)

multilayer thin-film ceramic, and silicon-on-silicon superhybrids). ASP remains independent of technology, so it can benefit from the inevitable improvement in microelectronics technology without architectural modification.

ASP system architecture

As indicated in Figure 1, an ASP system comprises a dynamically reconfigurable parallel processing structure of communicating ASP substrings, each supported with an ASP data buffer (ADB), a controller, and a data communications network.

ASP substrings. Each ASP substring comprises a string of identical APEs (associative processing elements), as shown in Figure 2. Each APE connects to an inter-APE communications network (which runs in parallel with the APE string). All APEs share common bit-parallel data, activity, and control buses, and one feedback line called Match Reply, or MR. An external controller maintains the buses, feedback line, and Link

ASP

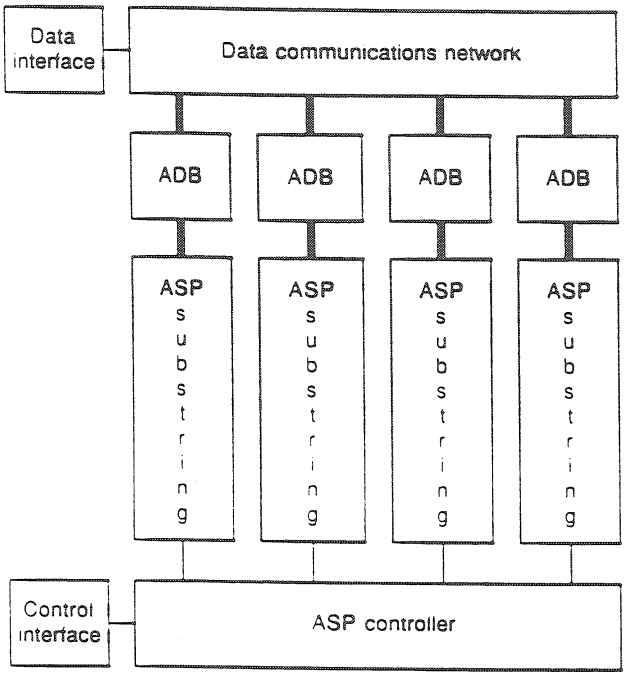


Figure 1. Activation of matching/mismatching APEs: before (1) and after (2).

Left and Link Right ports (LKL and LKR) of the inter-APE communications network.[2-4]

In contrast to more traditional parallel computer architectures, ASP uses content-matching rather than location-addressing techniques. Thus, ASP selects APEs for subsequent parallel processing by comparing their data and activity content with the states of the corresponding data and activity buses. Moreover, the lack of location addressing also simplifies system configuration and extension, implementation, and especially fault tolerance.

In operation, each ASP substring supports a form of set processing in which the subset of active APEs (those which match broadcast data and activity values) support scalar-vector and vector-vector operations. ASP either directly activates matching APEs or uses source inter-APE communications to indirectly activate other APEs (see the accompanying box). The match reply line indicates whether or not any APEs match. The controller either directly broadcasts scalar data or receives it via the bit-parallel data bus.

Similarly, ASP can also exchange input-output vector data (output dumped and input loaded in a single step) sequentially in APEs via the data bus with the bit-parallel primary data exchanger (PDX). As shown in Figure 2, the bit-parallel primary data exchanger

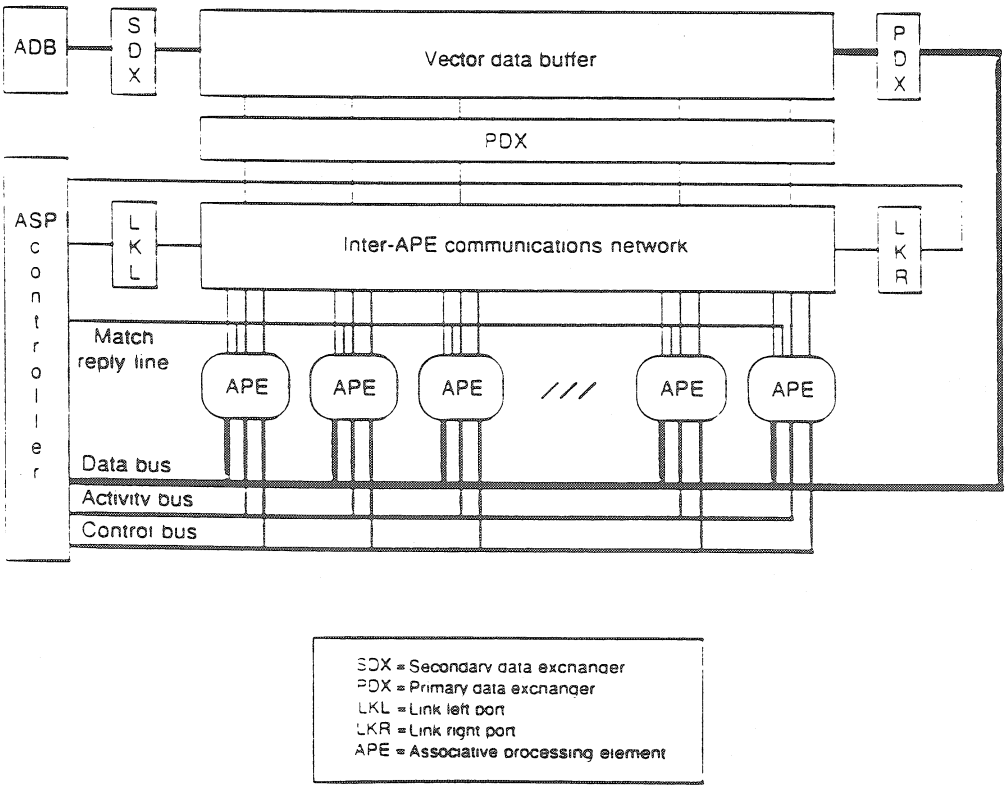


Figure 2. ASP substring.

APE Activation Options

Associative processing elements may be activated for read or write operations by one of the following activation options.

Matching and mismatching APEs

Assuming the state of an ASP substring shown in Figure A1, where M represents matching APEs, activation A of matching and mismatching APEs appears as seen in Figure A2.

Asynchronous communication

For each of the five activation examples shown in Figure B, the first ASP substring state indicates matching M APEs. The second and third states indicate activations A following asynchronous signal transmission (to the left and right respectively) from LKL, LKR, or source S matching APEs to destination D previously matched APEs. The examples assume inter-APE communication within a single ASP segment, such that all block links are closed. As an option of the fourth and fifth activations, the first source APE may also be included in the set of activated APEs.

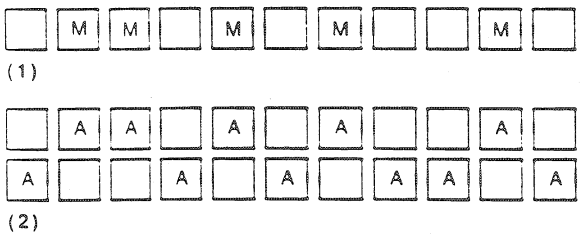


Figure A. Activation of matching/mismatching APEs: before (1) and after (2).

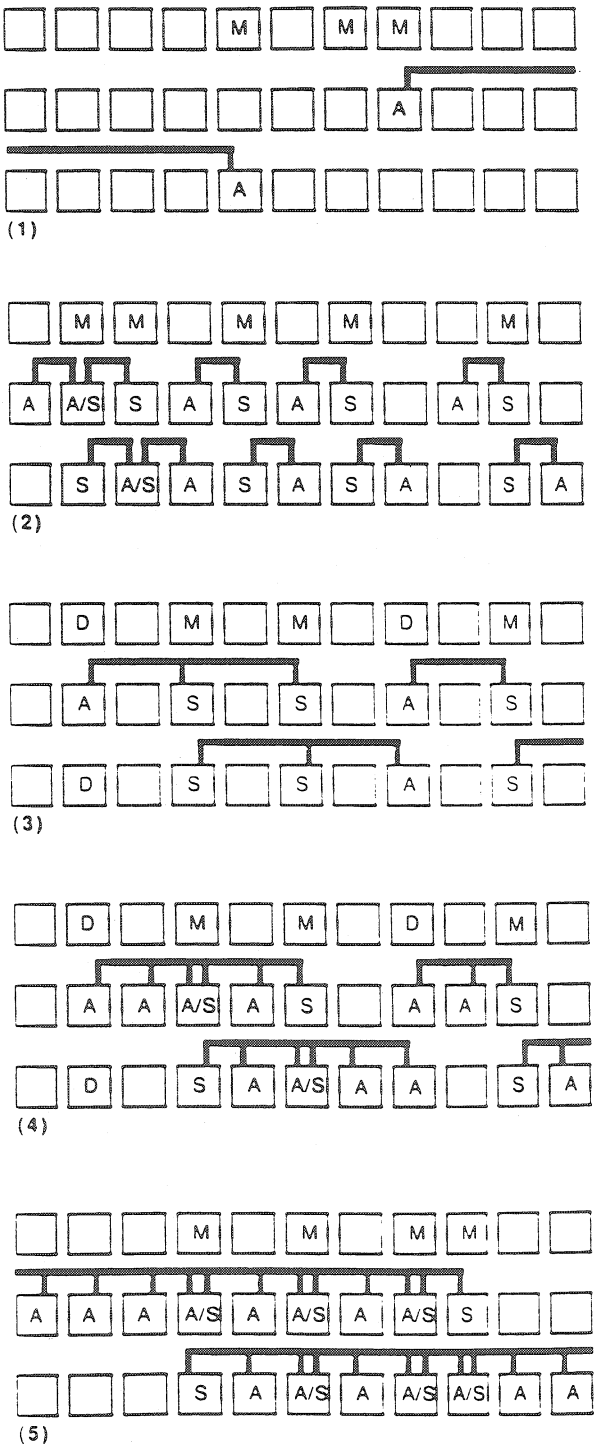


Figure B. Activations with asynchronous communication: an isolated matching APE (1); neighbors of matching APEs (2); remote APEs linked with matching APEs (3); substrings between matching APEs (4); and all APEs between matching APEs and one end of the string (5).

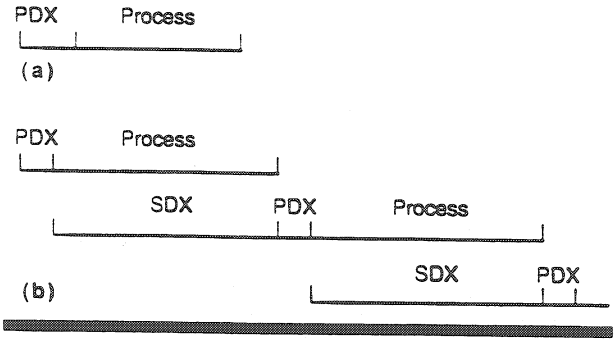


Figure 3. Example of unacceptable loss of efficiency (a); minimal loss of efficiency (b).

exchanges data. However, ASP loses the parallel processing advantage during sequential vector data exchange. Thus, depending on the time required to process the loaded vector data, the exchange could incur an unacceptable loss of parallel processing efficiency, as indicated in Figure 3a. In such cases, a vector data buffer supports a much faster APE-parallel exchange facility, in which a bit-serial PDX performs the task at a very high data rate. Thus we minimize the loss of parallel processing efficiency (Figure 3b).

Similarly, but at a lower data rate, the secondary data exchanger (SDX) provides a bit-parallel vector data exchange between the vector data buffer and the external ADB. The SDX overlaps parallel processing and, therefore, does not present a sequential processing overhead. Consequently, whereas the bit-parallel PDX is a fundamental feature of a substring, the vector data buffer and its support bit-serial PDX and SDX are optional components of ASP. They are incorporated only for those applications requiring relatively short parallel processing periods.

Associative processing element. Each APE incorporates an n-bit data register and an a-bit activity register, an $(n+a)$-bit parallel comparator in which the values of n and a are 32 to 128 bits and 4 to 8 bits, depending on the application class for which ASP is optimized. Moreover, an APE includes a single-bit full-adder and four status flags (C to represent arithmetic carry, M and D to tag matching and destination APEs, and A to activate selected APEs). An APE also includes control logic for local processing and communication with other APEs. See Figure 4.

Data modes and activity bits. ASP hardware supports three modes of data representation (word, byte, and bit) within the n-bit data register (DR), as defined (in Pascal) in the following example (for n = 32 bits).

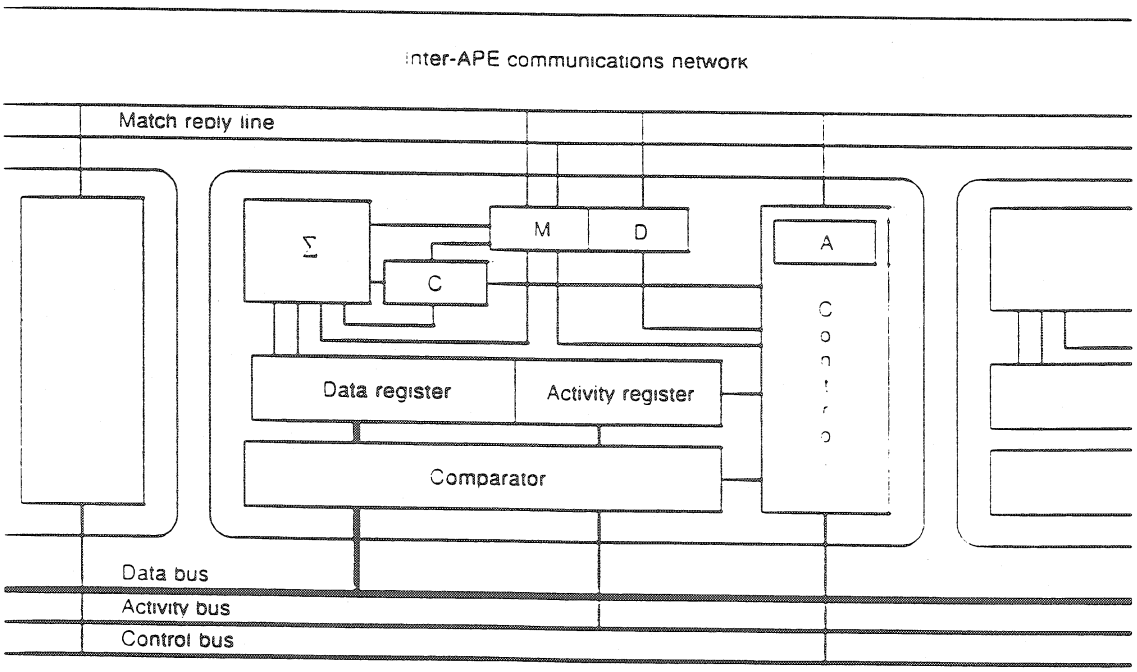


Figure 4. Associative processing element, or APE.

```
DR = record
       case data_mode of
         word_mode : (word : array [0..31] of 0..1);
         byte_mode : (byte : array [0..3,0..7] of 0..1);
         bit_mode  : (sf_1 : array [3..13] of 0..1;
                      sf_2 : array [14..22] of 0..1;
                      sf_3 : array [25..31] of 0..1)
       end
```

In word mode the data register provides storage for (and supports bit-parallel processing of) an n-bit binary word, as shown in Figure 5a. Alternatively, in byte mode the data register stores 8-bit byte fields, as shown in Figure 5b, and supports bit-parallel processing of a selected byte field. In bit mode the data register can store variable-length binary fields for bit-serial processing (Figure 5c). Users can declare one, two, or three such serial fields for unary, binary, or ternary bit-serial operations. Moreover, data representation is not limited by the n-bits of a single data register, since, in all three data modes, a contiguous string of APEs can be allocated for operand storage.

In contrast, the a-bit activity register provides storage for an a-element ordered set defined in Figure 5d (for 5 activity bits) as a Pascal-set type. Sets are stored such that the inclusion or exclusion of aB is represented by the state (1 or 0 respectively) of the Bth activity bit in the activity register.

Data and activity masking. To support data and activity masking, without the processing overheads normally incurred with specific mask registers, both the data bus and the activity bus support ternary data. The data bus incorporates a mask field, such that, for each bit of the data register, the data bus can support a 2-bit ternary digit representing one of three values (dX, d0, and d1).

The digits of the data bus in selected byte fields and those corresponding to the indexed bits of selected serial fields (in byte and bit modes) may be set to any of the three values. And, all digits in nonselected byte fields and serial fields, plus those digits corresponding to the nonindexed bits of selected serial fields, automatically assume the dX value. However, to economize on ASP chip input ports, designers chose not to allow the dX value in word mode.

Similarly, the activity bus supports 2-bit ternary digits such that its Bth activity digit can represent the presence or absence of aB or a masked aB in the activity register.

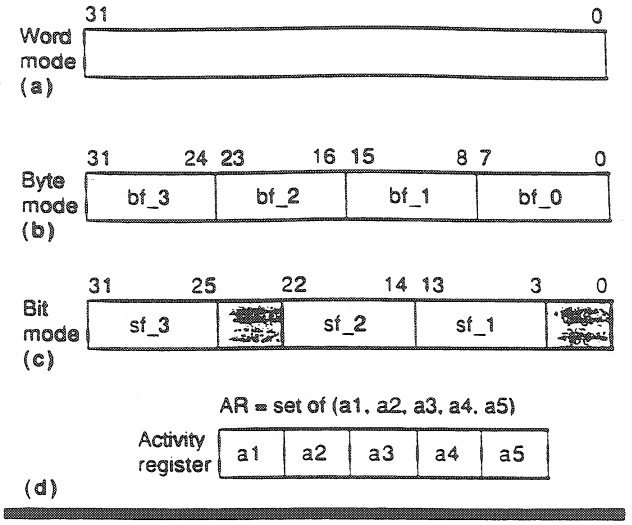


Figure 5. Data representation modes in the data register: word (a), byte (b), and bit (c). Activity register storage (d).

Basic APE operations. APEs support four basic operations, match, add, read, and write. Examples of each in pseudo-Pascal statements appear in the accompanying box. In a match operation the M and D tags become true in matching APEs and false in mismatching APEs. In an add operation the M tag and the C flag (in all APEs matching the specified activity bits) repre-

Basic APE Operations

Match: the M and D tags become true in matching APEs and false in mismatching APEs, as suggested by the following pseudo-Pascal statements

```
forall APEs do                                  {simultaneously in each APE}
  if DR__match and AR__match
    then case tagoption of
           M        : the M tag becomes true;
           MandD    : both the M and D tags become true;
           DaltM    : for each ASP segment, the D and M
                      tags become true starting (from the left
                      end of the segment) with D
         end
```

sent the sum and carry of a bit-serial addition (or subtraction) operation. During read, ASP updates each digit of the data bus DB[j] to the state of the wire-AND (0's are "stronger" than 1's) of the corresponding DR[j] bits of all activated APEs and their activity registers can be updated. During a write operation ASP updates the data register and the activity register in activated APEs according to the states of the data bus and the activity bus.

Inter-APE communications network. As indicated in Figure 2, each substring supports two styles of inter-APE communication,

- bit-parallel, single-APE communication via the shared data bus and
- bit-serial, multiple-APE communication via the inter-APE communications network.

Although the former can be used to advantage on many occasions, we discuss the latter here.

The inter-APE communications network implements a globally controlled and dynamically reconfigurable, tightly coupled APE interconnection strategy. This strategy supports cost-effective emulation of common network topologies (see box on p. 18). Most significantly, the APE interconnection strategy supports simple modular network extension, to enable tailoring of parallel processing power to match user requirements.

In contrast to the networks adopted by other parallel computer architectures, we did not design the inter-APE communications network primarily for the transfer of actual data between APEs. Instead, and much more simply, we restricted communication to the high-speed transfer of activity signals (or M-tag patterns) between neighboring or selected remote APEs (those matching the selection criteria). Since APEs can easily be activated by content addressing and their data content processed in situ, we reduce the time-consuming movement of data to an absolute minimum.

```
        else case tagoption of
               M              :  the M tag becomes false;
               MandD, DaltM   :  both the M and D tags become false
             end
```

where DR__match and AR__match are defined as follows.

For each APE, assuming DR__match has been initialized as true

```
forall j in [1..n] do                          {simultaneously for each digit}
  if DB[j] < > dX
    then DR__match := DR__match and (DB[j] = DR[j]);
AR__match := ([included AB bits] < = AR) and    {set inclusion test}
             (AR < = ([a1,a2,a3,a4,a5] - [excluded AB bits]))
```

If any M tag becomes true, as a result of the match operation, then the global Match Reply also becomes true, otherwise MR becomes false.

Add: in all APEs matching the specified activity bits, the M tag and C flag represent the sum and carry of a bit-serial addition (or subtraction) operation, as suggested by the following pseudo-Pascal statements, assuming C has been initialized as '0'

```
forall APEs do                                 {simultaneously in each APE}
  if AR__match
    then begin
           M := not M0 < > (M1 < > (C = 1));
           if not M0 and (M1 or (C = 1)) or (M1 and (C = 1))
             then C := 1
             else C := 0
         end
```

where, M0 and M1 are derived in each APE, for the bits (indexed by j and k) of selected serial-fields, as follows

```
case addend of
  scalar    : M0 := DB(J) = DR[z];              { scalar addend}
  vector    : M0 := DR(j) = 0;                  {vector addend}
  M__tag    : M0 := not M                       {vector addend}
end;
M1 := DR[k] = 1                                 {vector augend}
```

In fact, the APE interconnection strategy supports two modes of inter-APE communication:

• **Circuit-switching.** Asynchronous, bidirectional, single-bit communication occurs via multiple signal paths, dynamically configured (programmer transparent) to connect APE sources and corresponding APE destinations of high-speed activation signals. Circuit-switching implements a fully connected permutation and broadcast network for APE selection and inter-APE routing functions.

• **Packet-switching.** Synchronous, bidirectional, multibit communication via a high-speed bit-serial shift register, routing M-tag patterns along each APE substring for data/message transfer.

To preserve continuity at the two ends of the inter-APE communications network, the LKL and LKR ports (shown in Figure 2) allow activation or M-bit signals to be injected and sensed by the external ASP controller. They also act as the left and right neighbors of the leftmost and rightmost APEs in the associative string processor.

APE block bypassing. Each substring is partitioned into equal-length APE blocks, separated by block links, as indicated in Figure 6 on p. 19. At an abstract level, and assuming a programmable connection between the LKL and LKR ports, the inter-APE communications network can be considered as a hierarchical chordal-ring structure, with the chords bypassing APE blocks (and groups of APE blocks). The network:

• *Accelerates inter-APE communication signals.* APE blocks, not including destination APEs are automatically bypassed for both circuit-switched and (if required) packet-switched modes of inter-APE communication and, if appropriate for the former mode, activated in a single step.

where DR(z) indexes a single-bit serial-field of 0's, and for subtraction (i.e., 2's complement addition), C is initialized as '1' and M0 is not complemented.

Read: each digit of the data bus DB[j] is updated to the state of the wire-AND (i.e., 0's are "stronger" than 1's) of the corresponding DR[j] bits of all activated APEs, as suggested by the pseudo-Pascal statements

```
forall j in [1..n] do                          {simultaneously for each bit}
  if DR[j]__test
    then DB[j] := d0
    else DB[j] := d1
```

where DR[j]__test, assuming it has been initialized as false, is derived as follows

```
forall APEs do                                 {simultaneously in each APE}
  if A
    then DR[j]__test := DR[j]__test or (DR[j] = 0)
```

and the Activity Register AR is updated as for write operations.

Clearly, it is sensible to activate only one APE per ASP substring for read operations.

Write: the Data Register (DR) and Activity Register (AR) are updated in activated APEs according to the states of the data bus and the activity bus as suggested by the following pseudo-Pascal statements

```
forall APEs do                                 {simultaneously in each APE}
  if A
    then begin
           forall j in [1..n] do               {simultaneously for each bit}
             if DB[j] <> dX
               then DR[j] := DB[j];
         AR := AR + [included AB bits] - [excluded AB bits]
       end                                     {set union and difference}
```

ASP Support of Structured Data

We specifically designed the reconfigurable inter-APE communications network of the ASP substring to support any data structure with cost-effective emulation of common network topologies. Examples of these topologies appear in Figures C through F.

Arrays

For vectors, data registers store consecutive elements j in consecutive APEs within a segment, with an activity bit or L's, providing local origins for element j indexing. See Figure C1.

For matrices, concatenated vectors (segments) separated with an activity bit or L's represent consecutive matrix rows i. L's provide local origins for column j indexing, as indicated in Figure C2.

For cubes, hyperspace structures can be represented in a similar manner to that shown for the matrix. For example, the binary n-cube (where n = 3) structure could be represented as shown in Figure C3 and navigated with exchange mappings (see address permutations).

Tables

The fields F, each being allocated one or more consecutive APEs, are ordered and delimited with an activity bit (for example, a1). Similarly, entries can be separated with another activity bit or with L's, if each entry is allocated a segment (see Figure D).

Trees

An n-ary tree can be represented in list form, with the head of each sublist being a parent node and the tail comprising the n-child nodes. One or more consecutive APEs represent each node. Sublists may be delimited with an activity bit or with parentheses as:

(A (B C D) (E F (G H I)))

With this representation many subtrees can be navigated and reduced in parallel.

Graphs

Semantic networks may be represented in list form, as indicated earlier for trees. However, for large complex networks, APE data registers can be allocated to nodes, which comprise in-links (lowercase letters), data (uppercase letters), and out-links (see Figure E). Users navigate the semantic network by searching for an in-link that has been read from the out-links of the previously matching node.

In general, for address permutation networks we try to avoid the actual movement of data between

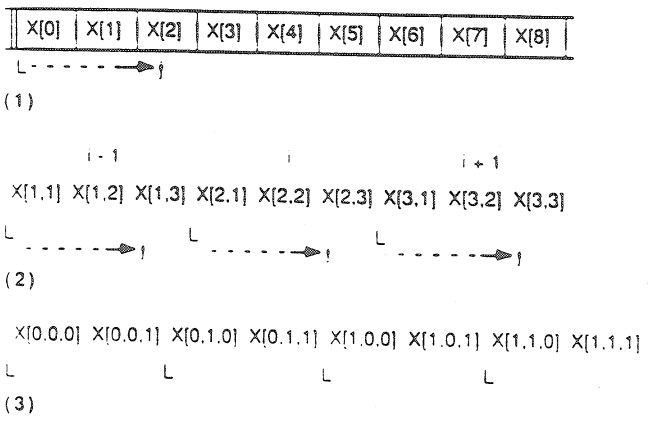


Figure C. Array examples for vector (1); matrix (2); and cube (3).

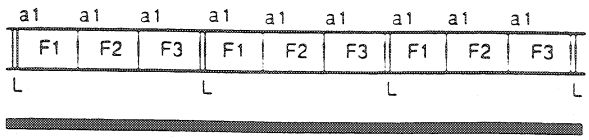


Figure D. Table example.

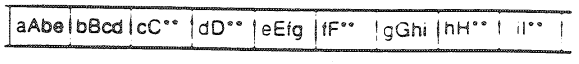


Figure E. Tree example.

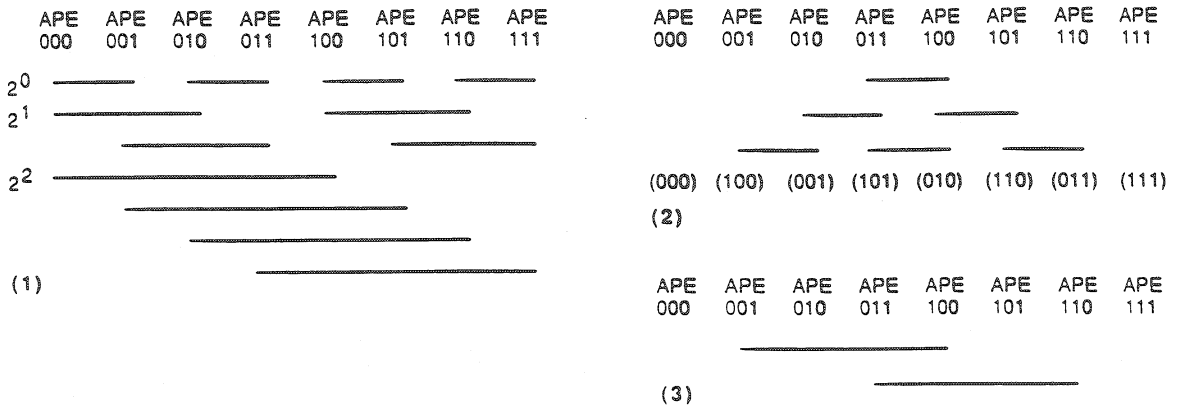


Figure F. Moving data between APEs by exchange (1), shuffle (2), and butterfly (3) techniques.

physically addressed APEs in favor of content addressing and activation alternatives. Modifying logical addresses assigned to a field of APE data registers effectively achieves data transfer, for those algorithms requiring association of data with processor location. However, if data movement between physically addressed APEs is definitely required, we accomplish it with one or more of the following techniques.

- **Exchange.** We achieve a specified mapping with a single shift in each direction along the substring for a distance that is an appropriate power of 2, as indicated in Figure F1's three examples.
- **Shuffle.** For N APEs, we achieve the perfect shuffle in $\log_2 N$ steps by pyramidal neighbor exchange, as shown in Figure F2.
- **Butterfly.** For N APEs, a single exchange, as indicated in Figure F3, achieves this address permutation.
- **Shifting.** As its name suggests, we achieve this address permutation by synchronously shifting (left or right) M-tag patterns between selected APEs.

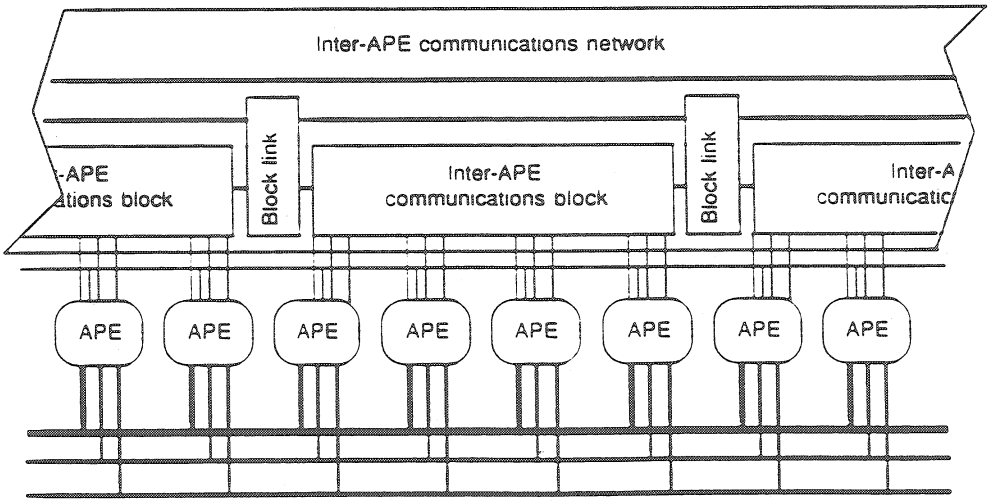


Figure 6. APE block bypassing.

• *Provides APE block defect/fault tolerance.* APE blocks failing a test routine, either in manufacture or service, switch out of the string, such that defective or faulty blocks are simply bypassed.

To provide cost-effective improvement of inter-APE communication speed and defect/fault tolerance, APE block bypassing depends on the implementation. For example, a VLSI ASP may support bypassing of 8-, 64-, and 256-APE block groups and entire ASP substrings. ULSI/WSI ASPs bypassing block groups of 4 and 64 APEs and entire branches (ASP substrings) may be specified as design criteria.

To avoid loss of computational efficiency, designers chose to make hardware features of the APE blocks, block links, and block bypassing transparent to the ASP programmer.

ASP segments and segment links. Each substring may be partitioned into programmer-defined segments, separated by segment links, in support of structured data. See box entitled "ASP Support of Structured data."APE block links incorporate bistables, which can be toggled to convert the block links into programmable segment links. The segment links can be opened or closed to prevent or allow the transmission of inter-APE communication signals between adjacent segments. Thus, each segment comprises a span of contiguous APE blocks, with internal block links closed and end block links converted to segment links.

Users can create variable-length segments by writing segment links corresponding to the M_tags at the ends of APE blocks. Alternatively, they can create equal-length segments, comprising power-of-2 APEs, with a special command.

ASP data buffer. Each ADB module provides data block storage immediately before and after processing within its local substring as shown in Figure 1. In addition to data storage each ADB module incorporates bit-parallel primary and secondary data exchangers (PDX and SDX) similar to those shown for the vector data buffer in Figure 2. In operation the PDX of each ADB module complements the SDX of its ASP substring. The corresponding SDX in each ADB module complements the data interface (DI) and the SDX of other ADB modules via the data communication network.

Consequently, in contrast to inter-APE communication within each substring, the ADB modules allow data communication between substrings to be fully overlapped with substring processing. This feature avoids data transfer overheads.

ASP controller. With appropriate management of the LKL and LKR ports of multiple ASP substrings within the ASP controller, we achieve two different configurations:

• multiple substrings, under the control of one controller, to form a SIMSIMD (single-instruction control of multiple SIMD modules) configuration, and

• multiple substrings, each under the control of an independent controller, to form a MIMSIMD (multiple-instruction control of multiple SIMD modules) configuration.

Moreover, by feeding LKL and LKR ports to a programmable router (within the ASP controller), users could also construct reconfigurable ASP structures that could adapt dynamically to the changing needs of complex computational tasks. The reconfigurable structures would provide flexible coverage of the wide range of information processing applications already outlined.

In operation, the ASP controller executes a sequence of procedures (stored as microprograms) called by an application program running in the host machine. All APEs in a substring receive microinstructions that are broadcast on the control bus. In addition to the microprogram unit, the ASP controller also incorporates a scalar data buffer and a scalar processor.

In common with typical high-speed microprogram-controllers, users could implement a general-purpose controller with standard bit-slice microprocessor components on two double-extended Eurocards. This approach could be reduced to one board for simpler controllers. Indeed, in cases where eventual production volume justifies the extra development cost, semi-custom VLSI implementation could reduce the controller to a small chip set. Moreover, for some application-specific ASPs, dedicated control logic could be incorporated on the VLSI ASP substring chip.

ASP data communications network. The interprocessor communication network usually dominates parallel computer cost and performance. Consequently, the integration of a cost-effective APE interconnection strategy was a major design goal for the ASP system architecture.

Investigation of the interprocessor communication requirements for a wide range of parallel algorithms revealed a Zipf-curve relationship between frequency and distance. Short-distance communication is common and long-distance communication is rare. Consequently, the hierarchical interconnection strategy adopted for ASP supports a high degree of parallelism for local communication (within the inter-APE communications network) and progressively lower degrees of parallelism for longer distance communication (within the inter-APE communications and data communications networks).

Thus, we do not restrict SIMSIMD and MIMSIMD configurations to the chordal-ring structure of the inter-APE communications network. For example, we could configure the data communications network

ASP Operations

In operation, the ASP controller dynamically formats the data bus and, consequently, the data registers of all APEs to support appropriate operand fields (see Figures G1 and G2).

ASP performs parallel operations on selected APE subsets, as indicated by the statements (ASP software) shown in Figure H. Such operations support both bit-serial and bit-parallel operations.

Bit-serial/word-parallel

Assuming the data register format shown here, ASP supports bit-serial/word-parallel, scalar-vector (S-V) and vector-vector (V-V) arithmetic, relational, and logical operations (o) as seen in Figure I.

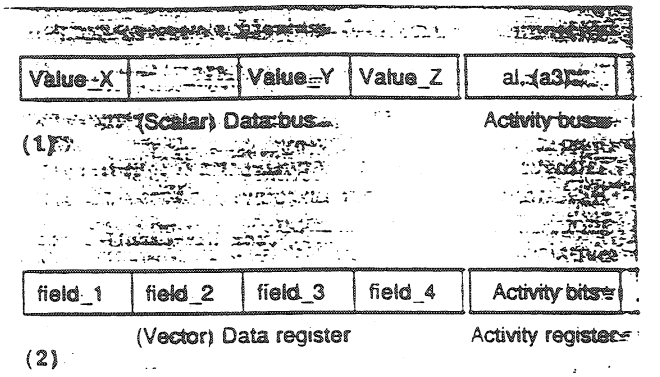


Figure G. Operation examples: scalar data bus (1); vector data register (2).

```
where <scalar-vector match condition> tag <tag-option>;
activate <activation option> do
    <scalar-vector or vector-vector operation>
```

Figure H. ASP operation statement.

```
e.g.  for j := ls_bit to ms_bit do
        begin
          where (DR.field_1 = Value_X) and
                (a1 in AR) and not (a3 in AR) tag M;
          activate M_tagged_APEs do
            begin
              DR.field_4[j] := Value_Y[j] o DR.field_3[j];     (S-V)
              DR.field_3[j] := DR.field_1[j] o DR.field_2[j]   (V-V)
            end
        end
```

Figure I. Bit-serial/word-parallel operations.

shown in Figure I to implement an alternative network topology (cross-bar, mesh, torus, shuffle, exchange, butterfly, or binary n-cube).

In summary, to optimize cost-effectiveness and application flexibility, ASP incorporates two levels of interprocessor communication. The lower level provides low-cost implementation of large-scale, fine-grain parallelism, and the upper level minimizes (what would have been a high) implementation cost with small-scale, medium-grain parallelism. In addition, a particular performance advantage of this two-level interconnection strategy is that upper-level communications can be fully overlapped with substring processing. To illustrate the functionality of the architecture we've just described, see ASP operations box.

ASP software

The degree of programmability and associated software complexity are key issues for general-purpose parallel computer design. Unfortunately, the history of parallel algorithm development reveals a profoundly steep learning curve, with much evidence of poor exploitation of natural parallelism and cost-ineffective use of applied parallelism. The height of the learning

```
        where DR.field_1 = value_X tag M;                    {byte mode only}
        activate M_tagged_APEs do
          begin
            DR.field_4 := Value_Z;                           {byte mode only}
            AR := AR + [a1] - [a3]                  {set union and difference}
(1)       end


        e.g.  for bit-parallel vector-vector addition of APEs  marked with
              activity bit a4, assuming the prior execution of

                where a4 in AR tag M;
                activate M_tagged_APEs do C := 0

                where (DR[j] = 0) and (DR[k] = 0)
                      and (a4 in AR) tag MandD;
                where (DR[j] = 1) and (DR[k] = 1) and (a4 in AR) tag M;
                activate left_substrings_of_M_tagged_APEs do C := 1;
                where a4 in AR tag M;
                activate M_tagged_APEs do DR[r] := DR[j] + DR[k]

(2)           where the variables r, j and k support word-indexing.
```

Figure J. Scalar-vector matching and assignment operations (1); vector-vector arithmetic, relational and logical operations (2).

Bit-parallel/word-parallel

ASP executes scalar-vector and vector-vector operations in bit-parallel modes, as outlined in Figures J1 and J2. Assuming our data register format, ASP supports bit-parallel/word-parallel, scalar-vector matching and assignment operations of the type seen in Figure J1.

Assuming that R-bit data registers of a group of P contiguous APEs are allocated to store R P-bit data words, ASP also supports bit-parallel/word-parallel vector-vector arithmetic, relational, and logical operations. See Figure J2. Note that in this operating mode we configure an N-APE ASP as N/P P-bit processors, each supported by R P-bit registers. For example, the 256-APE VLSI chip shown in Figure 9 could support 32 8-bit (or sixteen 16-bit or eight 32-bit or four 64-bit) microprocessors, each with 32 general-purpose registers in this mode.

curve may span as much as two to three orders of magnitude in performance.

The solution to this problem should be found in a high-level parallel programming language that offers simple expression of natural parallelism and flexible control of applied parallelism. But, as yet, no widely accepted such parallel programming language exists.

Experimenting with existing declarative languages and extending existing procedural languages with parallel processing constructs are unlikely to promote efficient parallel algorithms. Worse still, these languages do not encourage programmers to climb the learning curve and improve parallel processing performance. Indeed, algorithm performance benchmarks, published by parallel computer vendors offering such language extensions, are normally based on assembly-level coding.

Moreover, programmers are very reluctant to change their working environments. Indeed, the cultural change to parallel processing is traumatic enough, without complicating the issue by enforcing adoption of an alien programming language and (possibly) an unfamiliar operating system, not to mention the concomitant loss of access to existing software. Consequently, the pragmatic approach adopted for ASP accepts that parallel computer users are best served with their familiar (sequential) software development systems.

Users can write ASP application programs entirely in a familiar block-structured, high-level language (Pascal, Modula-2, C, or Ada) under a familiar operating system (Unix, VMS, or MS-DOS). Such programs include calls to external precompiled ASP algorithms and procedures, written and progressively refined

(in the same language) by experts, with an intimate knowledge of the ASP architecture, using a set of built-in function and procedure primitives. Thus, the average ASP application programmer/user does not face the full complexity of parallel algorithm development. The user has the much less-demanding task of selecting and interfacing appropriate code from the hierarchically organized ASP algorithm and procedure library. Only occasionally must programmers resort to the creation of such code.

Nevertheless, we are actively investigating parallel constructs for ASP programming. I offer the syntax definitions in Figure 7, written in EBNF (extended Backus-Naur formalism) and the example procedures in Figure 8 in an attempt to explain the nature of ASP programs.

ASP procedure examples. The first example of an ASP procedure marks the maximum and minimum values in the data field [ls_bit...ms_bit] of a subset of positive (binary) integers (already marked with activity bit a4) with activity bits a1 and a2. See Figure 8a.

The second example procedure in Figure 8b marks the first and last characters of all words (text strings delimited at both ends with spaces) that match an *n*-character word with activity bits a1 and a2 respectively.

Development program

As mentioned earlier, the ASP concept is particularly well matched to the exciting opportunities and exacting constraints of VLSI chip fabrication. Reasons include the high APE packing density, the highly compact inter-APE communications network, and, especially, the independence of I/O requirement from string length. (Compare the linear ASP with a two-dimensional array in which the I/O requirement grows as the square root of the array size.)

Moreover, ASP is highly amenable to defect/fault tolerance, owing to its construction from a large number of identical APEs, lack of location-dependent addressing and simple inter-APE interconnection. Consequently, as reducing feature sizes and increasing chip sizes drive VLSI chip fabrication technology toward the prospect of ULSI chips and WSI devices, the ASP architecture offers consistency and becomes increasingly more cost-effective.

Pioneering investigations. In the early seventies Brunel University developed two experimental ASP prototypes. The first (funded by the United Kingdom Department of Industry) was based on LSI associative memory chips, designed by the author and fabricated by GEC-Marconi. We based the second (funded by the UK Science and Engineering Research Council) on a TTL (transistor-transistor logic) emulation of an LSI ASP chip design. Both prototypes demonstrated the architectural principles and low-level software and stimulated interest in the application potential of ASP.

The ASP chip development program at Brunel University from 1976 until 1981 included the design at the university and the fabrication at Plessey of two LSI ASP test chips. The project demonstrated the feasibility of microelectronic implementation of the architecture.

In addition, a series of research contracts, funded by British Aerospace in 1978 strongly influenced ASP systems and software development. The projects investigated the application of ASP to real-time image processing tasks.

VLSI ASP chips. Since 1981 a three-phase program to develop VLSI ASP chips has been running at Brunel University.

Phase 1, experimental prototyping. Complementary SCAPE and Script projects ran from 1981 to 1987. Their objectives were to develop VLSI ASP chips that

```
statement = Pascal_statement ¦ ASP_construct.

ASP_construct = tag_statement ";" {activation_statement}.

tag_statement = ifany_statement ¦ where_statement.

ifany_statement = "ifany" tag_function "then" statement
                  ["else" statement].

where_statement = "where" tag_function.

tag_function  = match_condition "tag" tag_option.

activation_statement  = "activate" activation_option "do" APE_operation.
```

Figure 7. Syntax definition in ASP parallel constructs.

```
type nrange = 1..n;

ASP_procedure MAXMIN (ls_bit,ms_bit:nrange);
var j : integer;
begin {MAXMIN}
  where a4 in AR tag M;
  activate M_tagged_APES do AR := AR + [a1,a2];                    {set union}
  for j := ms_bit downto ls_bit do
    begin
      ifany (DR[j] = 1) and (a1 in AR) tag M
        then begin
               where (DR[j] = 0) and (a1 in AR) tag M;
               activate M_tagged_APEs do AR := AR - [a1]  {set difference}
             end
       ifany (DR[j] = 0) and (a2 in AR) tag M
        then begin
               where (DR[j] = 1) and (a2 in AR) tag M;
               activate M_tagged_APEs do AR := AR - [a2]  {set difference}
             end
    end
end {MAXMIN}
```

(a)

```
ASP_procedure FIND_WORDS (n:integer;word:string);
var i : integer;
    match : boolean;
begin {FIND_WORDS}
  i := 1; match := true;
  where [] <= AR tag M;
  activate M_tagged_APEs do AR := [];
  where DR = ' ' tag MandD;
  activate right_neighbours_of_M_tagged_APEs do
    AR := AR + [a1,a2];                                            {set union}
  while (i <= n) and match do
    begin
      ifany (DR = word[i]) and (a2 in AR) tag M
        then begin
               activate all_APEs do AR := AR - [a2];       {set difference}
               activate right_neighbours_of_M_tagged_APEs do
                 AR := AR + [a2]                                   {set union}
             end
        else match := false;
      i := i + 1
    end;
  if match
    then begin
           where (DR = ' ') and (a2 in AR) tag M;
           activate left_substrings_of_M_tagged_APEs do
             AR := AR + [a3];                                      {set union}
           where (DR = word[1]) and ([a1,a3] in AR) tag M;
           activate all_APEs do AR := AR - [a1,a3];        {set difference}
           activate M_tagged_APEs do AR := AR + [a1]             {set union}
           where a2 in AR tag M;
           activate all_APEs do AR := AR - [a2];           {set difference}
           activate left_neighbours_of_M_tagged_APEs do
             AR := AR + [a2]                                       {set union}
         end
    else write ('match fails')
end {FIND_WORDS}
```

(b)

Figure 8. Example ASP procedures: marking the maximum and minimum values in the data field (a); marking the first and last characters of all matching words (b).

Plans call for construction of application-specific and general-purpose chips for research and commercial exploitation.

would demonstrate the applicability of the ASP architecture to numerical (image processing) and nonnumerical (text-based symbolic processing) applications and assess their cost-effectiveness.

In 1986 Plessey fabricated the first samples of the 68-pin SCAPE (single-chip array processing element) chip. The chip is a 256-APE (32-bit data and five activity bits) VLSI ASP (funded by the UK Ministry of Defence from December 1982) in a 2μm complementary metal-oxide semiconductor (CMOS) process with two-layer metal.[4-7] The SCAPE chip appears in Figure 9.

A two-year research contract funded by UK's Alvey (Man Machine Interface) initiative involved collaboration with Quantel UK and the University of Bristol to investigate the design and evaluation of SCAPE-based image processing equipment.

Another two-year Alvey VLSI contract, starting November 1984 and involving Plessey, detailed a design requirement analysis and architectural specification for the Script chip. We based the chip on design and performance data derived from the SCAPE project.

Presently, Brunel University is investigating the design and evaluation of Script-based systems as part of the Scantrax (a back-end processor for relational database management and information retrieval) project.[8]

Phase 2, design consolidation. Since 1987 we based ASP implementation activity on the development of a cell-based design methodology. The methodology allows the VLSI chip designer to select fully engineered, exactly butting, CMOS layout cells from the ASP cell library and compose specific layout blocks for full-custom VLSI chips.

Both Brunel University and Aspex Microsystems staffs pursue such design consolidation in complementary tasks. The former activity investigates new cells and cell variants in research projects, and the latter provides the engineering development required to enable commercial VLSI ASP chips to be based on library cells.

To date, Phase 2 has involved the development of four VLSI ASP test chips, fabricated through the silicon foundry services of the UK's MCE Company

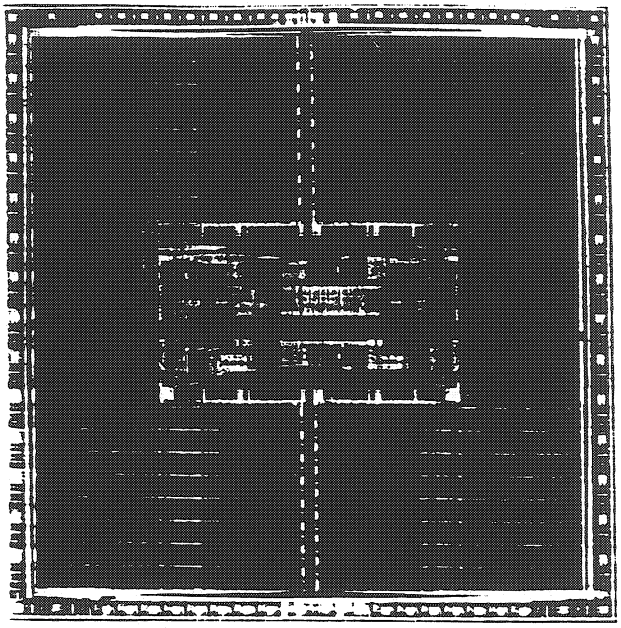


Figure 9. Photomicrograph of the VLSI ASP (SCAPE) chip.

and a 64-APE (32-bit data and five activity bits) VLSI chip. Currently, Plessey is fabricating the latter (in 2μm CMOS with two-layer metal). Based on new cells and incorporating design improvements, this SCAPE-like VLSI chip is intended for ASP demonstrator construction.

Phase 3, product development. Following establishment of a comprehensive cell library, plans call for phase 3 to develop both application-specific and general-purpose VLSI chips for ASP construction for both university research and commercial exploitation. Prominent among such chips are 256-APE and 1,024-APE VLSI chips being developed by Aspex Microsystems for fabrication in 1989 and 1990.

ULSI and WSI devices. Since the cost-performance potential of the ASP architecture improves with increasing string length, demonstration of the advantages of SCAPE-based image processing modules stimulated us to further research leading toward more improvements in cost-effectiveness. Moreover, since the ASP is so highly amenable to defect tolerance that chip area ceases to be a limiting factor, ULSI and WSI ASP devices become natural targets.

An important advantage gained by significantly increasing chip size is that we can also integrate much of the ASP controller and data communications network on the same chip. Indeed, in contrast to VLSI building blocks for the implementation of substrings (see Figures 2 and 9), ULSI and WSI devices implement an entire ASP system, as shown in Figure 1, on a single

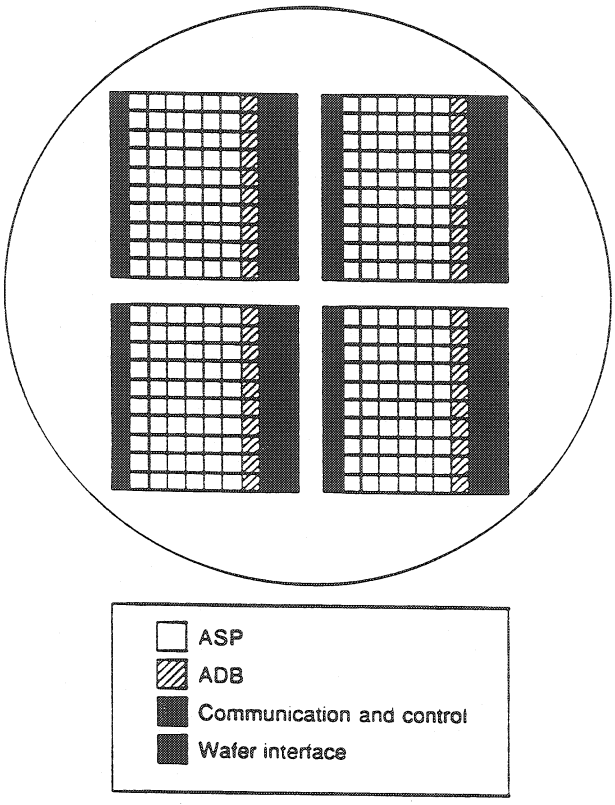


Figure 10. ULSI ASP chips.

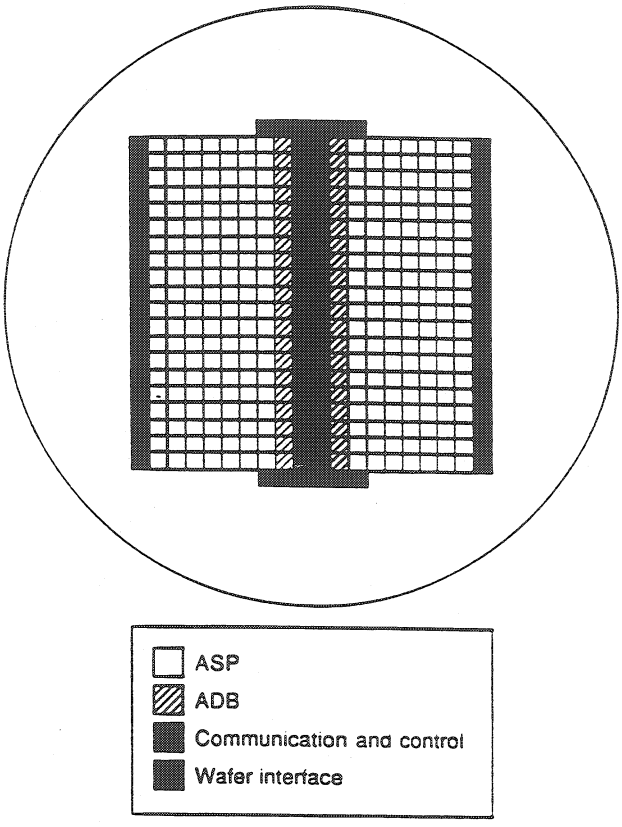


Figure 11. WSI ASP device.

silicon die. A single die offers major savings in size, weight, and cost together with increased reliability and ease of maintenance. ULSI and WSI devices, comprising 2,048 and 8,192 APEs, as shown in Figures 10 and 11, are being investigated in the WASP (WSI Associative String Processor) project at Brunel University.[8] The project includes the fabrication and evaluation of defect/fault-tolerant test chips and ULSI/WSI ASP technology demonstrators. It has been funded since 1985 under a UK Alvey (VLSI) contract and involves Plessey, GEC, ICL, and Middlesex Polytechnic.

In a separate project, Aspex Microsystems under a US Office of Naval Research contract is developing a WASP application demonstrator for iconic to symbolic image processing for fabrication in 1990.

ASP performance forecasts

To provide a simple indication of ASP performance, we designed VLSI/ULSI/WSI ASP chips (with 2μm CMOS fabrication technology) to allow each step of bit-serial and bit-parallel operations to be completed in 100 nanoseconds. Consequently, we can estimate the

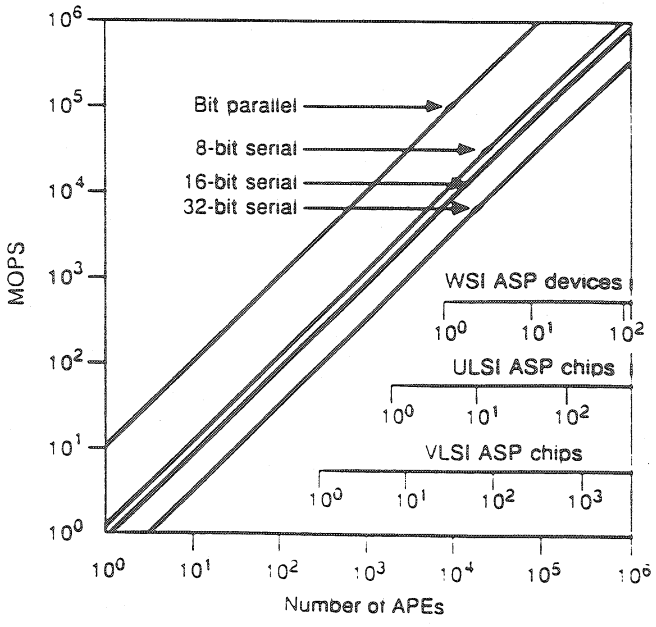


Figure 12. ASP performance estimation. MOPS = million operations per second.

limit of performance of an N-APE ASP approximately as follows:

$$\text{Performance} = \frac{10 \times N}{\text{No. steps per operation}} \text{MOPS}$$

Based on this expression, Figure 12 displays the potential performance of VLSI/ULSI ASP chips and WSI ASP devices.

The Associative String Processor is a highly versatile, parallel processing, computational architecture with the potential to achieve step functions in cost-performance over a wide range of information processing tasks. Indeed, ASP hardware and software modules provide the core technology for the construction of cost-effective, general-purpose and application-specific computer workstations designed for the end user.

Having described its architectural philosophy, structural organization, operational principles, and microelectronic implementation, I review the ASP in terms of the desirable features for fifth-generation computer systems shown in Table 2.

In terms of cost-effectiveness, application studies and benchmark evaluations demonstrate that ASP can match and often improve on the performance figures of contemporary parallel computers. And, consequently, the competitive edge becomes that of implementation cost.

Table 2.
ASP features applicable to fifth-generation computing systems.

Feature	Description
Application flexibility	As a fully programmable parallel processor with a reconfigurable interprocessor communication network, ASP provides flexible support for structured data processing thereby amortizing procurement costs over a wide range of both numerical and nonnumerical information processing applications.
Operational simplicity	By mapping all data to intermediate string-processing form, which can be accessed associatively and is supported by a reconfigurable interprocessor communication network, ASP provides simple support of data-level application parallelism (concurrency occurring naturally in structured data and in the algorithmic requirement) and simple control of process and instruction-level parallelism.
Computational efficiency	By enabling close matching between the applied parallelism of its architecture and the natural parallelism of applications with the interprocessor communication network, and by reducing sequential processing overheads with elimination of redundant processing, processor addressing, and unnecessary data movement, ASP maximizes computational efficiency with content addressing and in-situ processing.
Architectural extensibility	As a homogeneous parallel computer, ASP allows scaling of processing power with simple modular adjustment of string length to match user performance requirements.
Size and weight	By exploiting the latest advances in the VLSI-to-ULSI-to-WSI technological trend and high-density system assembly techniques, users can implement ASP in a very compact form.
Power requirement	Since only active processors dissipate and redundant processing has been eliminated, state-of-the-art microelectronic implementation allows ASP to be designed for low power consumption.
System reliability	As a fault-tolerant parallel architecture exploiting microelectronics technology, ASP offers high reliability and ease of maintenance.
Technology independence	Owing to its simple homogeneous string structure, ASP benefits from the inevitable improvement in microelectronics technology (increasing chip size and reducing feature size) without architectural or software modification.

ASP

In contrast to its contemporaries, we specifically developed ASP as a silicon-efficient parallel architecture. We achieved cost reductions by maximizing processor packing density, incorporating the interprocessor communications network on chip, and using defect/tolerant circuit design.

With 9,472 bits of content-addressable storage on a VLSI chip (with a 2μm feature size), Figure 9 demonstrates that the processor packing density ASP achieves is not too far behind that achievable with static RAM chips. Since system designers seem not to be concerned that all (but one) RAM storage locations are idle at any given instant, whereas many (if not all) ASP processors can be simultaneously active, it can be claimed that ASP has achieved the goal of compensating for inevitable processor redundancy with implementation cost reduction.

The curves of Figure 12 indicate potential ASP cost-performance. Assume that VLSI chips could be produced (in volume) for less than $100 each, that the chips dominate system implementation costs (an unlikely event, but prediction of system implementation costs are beyond the scope of this article). Allow (therefore) an arbitrary factor of 10 for a contingency/profit margin. Also estimate an order-of-magnitude improvement with WSI ASP devices. With these assumptions the comparative cost-performance curves in Figure 13 indicate the potential of 100 MOPS/$1,000 and 1,000 MOPS/$1,000 for VLSI- and WSI-based ASPs. For a more objective assessment of ASP cost-effectiveness, consider the following comparison of ASP with Thinking Machine's CM-1 (the Connection Machine).[9]

The 16-processor-element CM-1 processor chip and the 256-APE SCAPE chip share a common 68-pin package. Therefore, it seems reasonable to assume that a CM-1 printed circuit board (accommodating 32 processor chips, supporting RAM, and "glue" logic) could implement 8,192 APEs, supporting ASP data buffer and glue logic. Moreover, our detailed design studies show that a SIMSIMD ASP controller and data communications network for an ASP system, as shown in Figure 1, would only require one such board. The CM-1 requires at least an additional 12 boards (assumed, since the processor boards account for more than 90 percent of CM-1 circuitry[9]) for interprocessor communication, control, and input-output interfacing.

In summary, a 64K-APE VLSI ASP system would require only nine boards and an equivalent WSI ASP would require only eight wafers. Compare this with the 140 boards required for the 64K-processor CM-1. Moreover, with a one-square-centimeter area, the CM-1 chip costs approximately twice as much as the 75-square-millimeter SCAPE chip,[5,6] for typical VLSI foundry fabrication defect densities. For a 32-bit addition, the peak performance of a 64K-APE ASP would be around 20 gigaoperations per second, compared with only one GOPS for the CM-1. The CM-1 dissipates 12 kW of power; the VLSI and WSI ASPs would dissipate less than 300W and 100W.

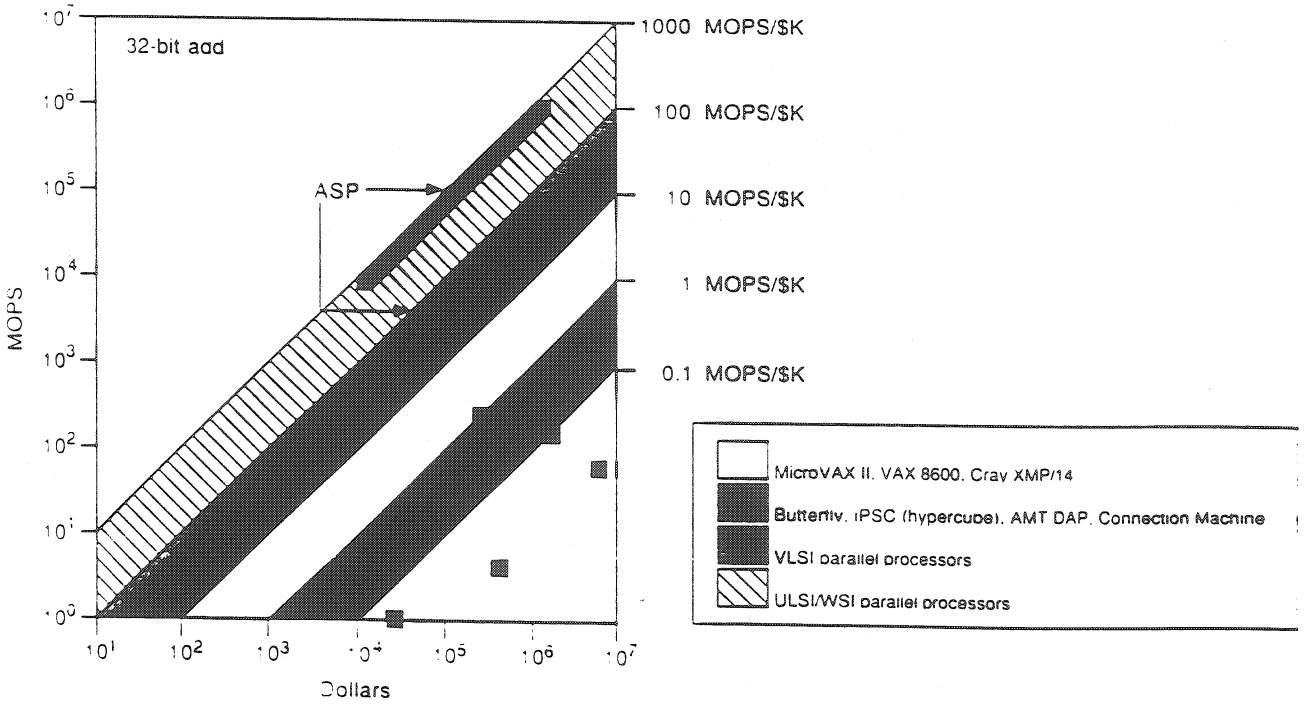


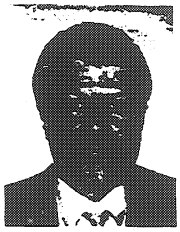
Figure 13. ASP cost-effectiveness. MOPS = million operations per second.

Acknowledgments

I gratefully acknowledge the enthusiastic contributions to ASP-based projects from past and present members of the Computer Architecture Group and Aspex Microsystems Ltd. at Brunel University, the support of the UK Alvey initiative, and US Office of Naval Research funding.

References

1. R.M. Lea, "Associative Processing," *Advanced Digital Information Systems*, I. Aleksander, ed., Prentice-Hall, New York, 1985, pp. 531-575.
2. R.M. Lea, "VLSI and WSI Associative String Processors for Cost-effective Parallel Processing," *The Computer Journal*, Vol. 29, No. 6, 1986, pp. 486-494.
3. R.M. Lea, "VLSI and WSI Associative String Processors for Structured Data Processing," *IEE Proc. Comput. and Digital Tech.*, London, Vol. 133, Pt. E3, 1986, pp. 153-162.
4. R.M. Lea, "The ASP, a Fault-Tolerant VLSI/ULSI/WSI Associative String Processor for Cost-Effective Systolic Processing," *Proc. IEEE Int'l. Conf. Systolic Arrays*, K. Bromley, S.Y. Kung, and E. Swartzlander, eds., CS Press, Los Alamitos, Calif., 1988, pp. 515-524.
5. R.M. Lea, "SCAPE: a Single-Chip Array Processing Element for Signal and Image Processing," *IEE Proc.*, Vol. 133, Pt. E3, 1986, pp. 145-151.
6. I.P. Jalowiecki and R.M. Lea, "A 256-Element Associative Parallel Processor," ISSCC, 1987, pp. 196-197.
7. S.R. Jones et al., "A 9Kbit Associative Memory for Parallel Processing Applications," *IEEE JSSC*, Vol. 23, No. 2, 1988, pp. 543-548.
8. R.M. Lea, "A WSI Image Processing Module, Wafer Scale Integration," G. Saucier and J. Trilhe, eds., Elsevier Science Publishers B.V. (North-Holland), 1986, pp. 43-58.
9. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, Mass., 1986.



R. M. Lea is the chief executive of Aspex Microsystems Ltd. and professor of microelectronics in the Electrical Engineering and Electronics Department of Brunel University. He specializes in parallel computer and VLSI chip architecture. In 1971 he began the Brunel Computer Architecture Research Group after recognizing the potential emerging from his GEC-Marconi associative memory project. While maintaining close links with industry, he has directed over 250 man-years of research in ASP systems for computer vision and symbolic-processing applications.

Lea founded Aspex Microsystems Ltd. in 1986 as a professional division of his research group to exploit ASP technology. Aspex complements academic research with industrial development and the ability to trade as an independent company.

Lea holds BSc and ARCS degrees from Imperial College and the MSc from Chelsea College, both of London University. He is an IEE fellow, and as speaker, chairman, and organizer he has contributed to many international workshops and conferences.

Questions concerning this article can be addressed to R. M. Lea, Brunel University, Uxbridge, Middlesex UB8 3PH, United Kingdom.