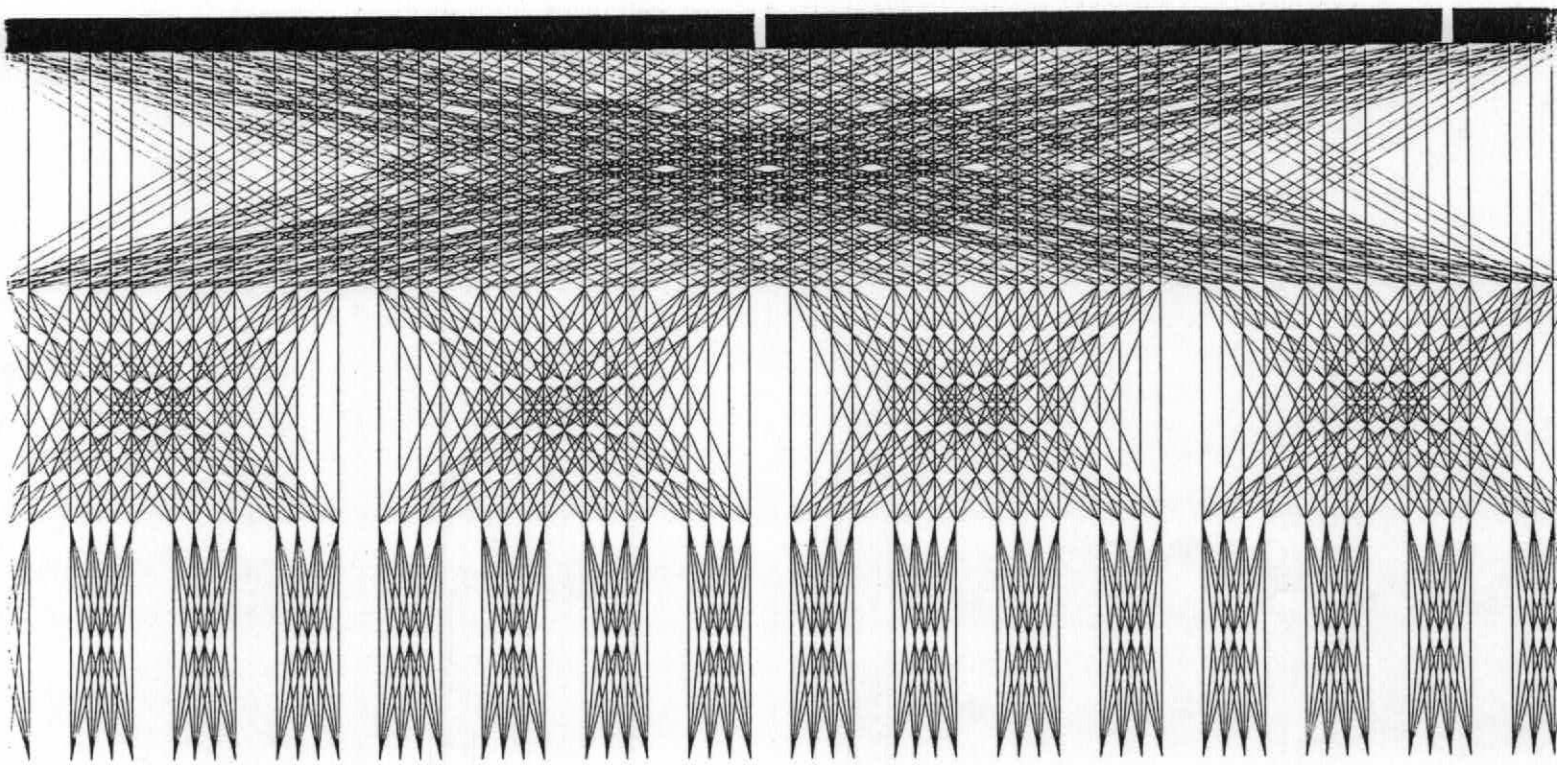
To: Bradley
From: CEL

recd BCH
12-2-92

CS-2



PRODUCT DESCRIPTION

meiko

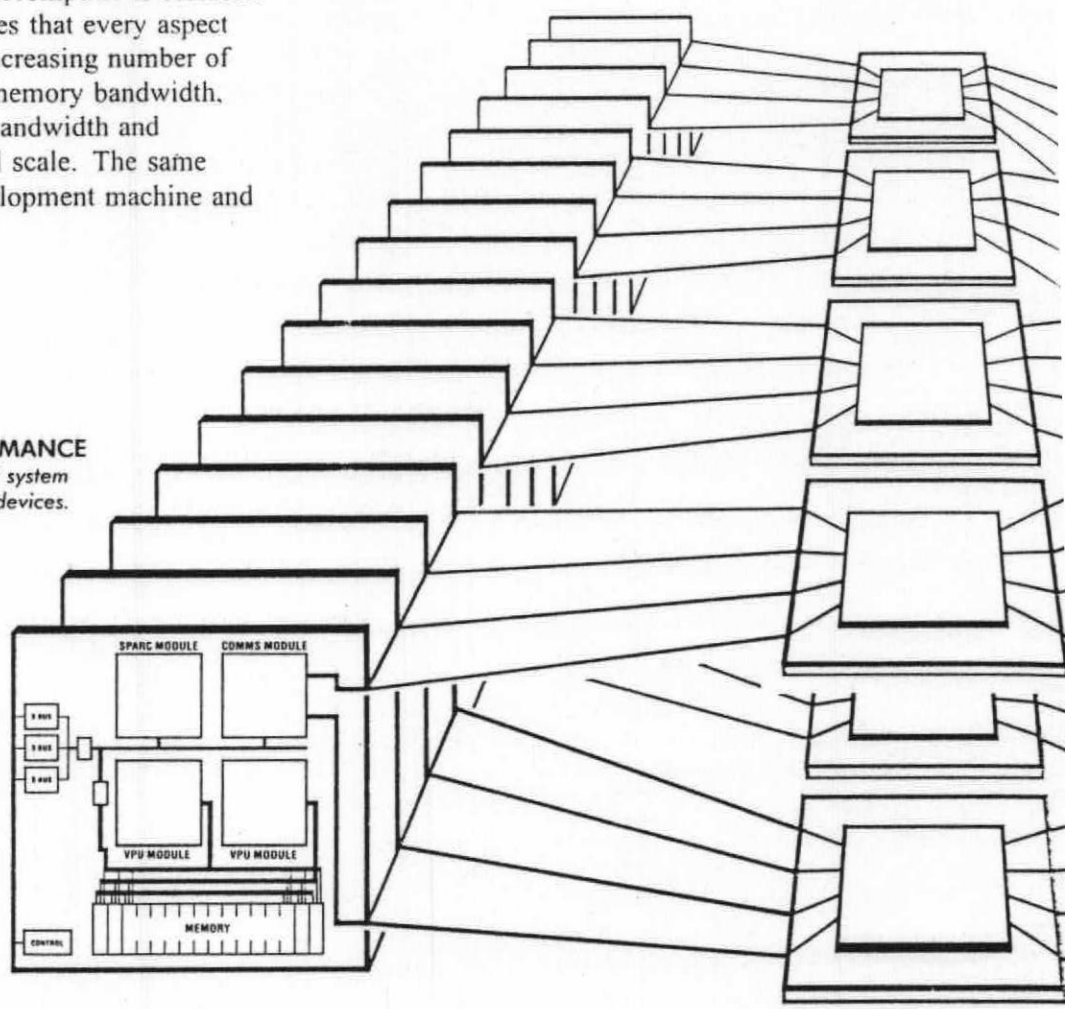
CS-2 introduces Seamless Supercomputing.
CS-2 is an MPP supercomputing system designed to focus more power on single problems than ever before. CS-2 is an MPP supercomputing system designed to support user communities in open systems environments more effectively than mainframes ever could or clusters of workstations ever will.

Every facet of the CS-2 supercomputer is scalable. Achieving true scalability requires that every aspect of the architecture scales with increasing number of processors. CPU performance, memory bandwidth, inter-processor communication bandwidth and I/O system performance must all scale. The same applications run on a small development machine and on a large production system.

SCALABLE PERFORMANCE
CS-2 performance scales linearly with increasing number of processors. Performance peaks at 200 Mflops per PE in 64-bit.

SCALABLE I/O PERFORMANCE
Every processor in a CS-2 system can manage its own I/O devices.

CS-2 PROCESSING ELEMENTS (PEs)
Every PE in a CS-2 system has one or more CPUs, its own local memory and a dedicated interface to the data network. CS-2 provides the option of scalar or vector PEs.



SWITCH STAGES
Every stage of the data network consists of multiple crossbar switches each connecting 8 data links.

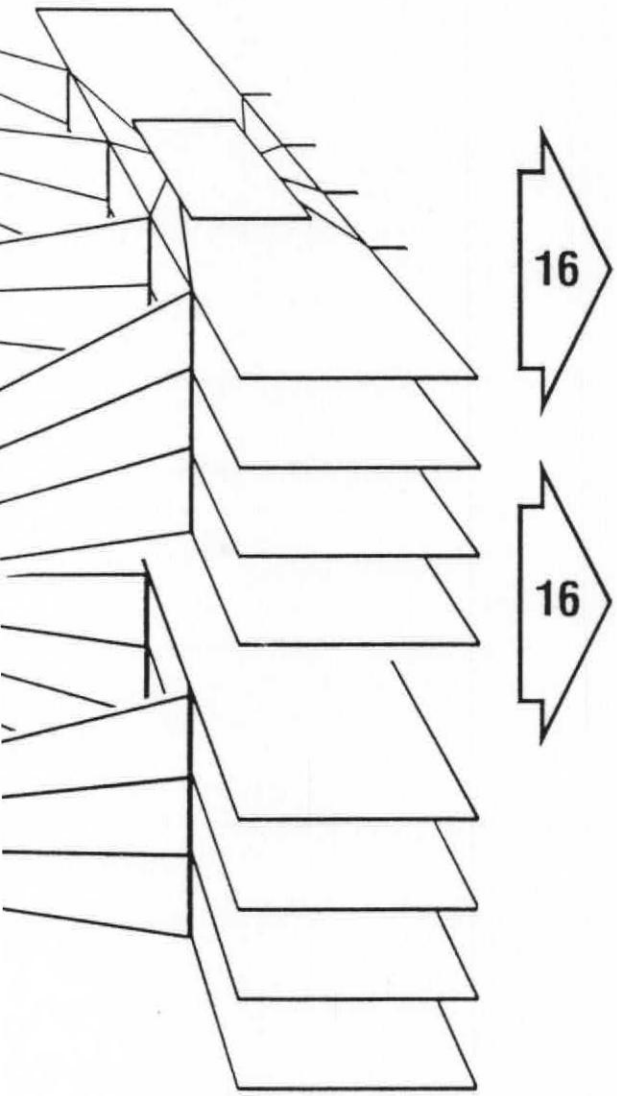
Scalable Processing

CS-2 is a distributed global memory architecture. Every processing element (PE) has one or more CPUs, its own local memory system and is capable of operating independently. The distributed memory architecture guarantees a constant ratio of cpu performance to memory bandwidth whatever the size of system – scalable cpu performance.

Scalable Inter-Processor Communication

Processors share data using a sophisticated and highly efficient communications network. Each PE has its own interface to this network, allowing it to access

SWITCH LAYERS
The CS-2 architecture supports multiple network layers. Each is capable of independent operation. A single layer provides multiple routes. Multiple layers increase bi-sectional bandwidth, reduce network contention and increase tolerance to failure.



CS-2 DATA NETWORK
Each layer of the CS-2 data network is a very fat tree with constant bandwidth per stage. One link per PE joins each stage of the network to the next.

data held anywhere in the system. The bi-sectional bandwidth of the CS-2 data network grows linearly with the number of PEs – truly scalable network performance.

The CS-2 data network is a multi-stage switch network; a fat tree with constant bandwidth per stage. As the number of PEs grows, network stages are added to preserve bandwidth. The data network provides scalable inter-processor communications performance with only logarithmically increasing complexity and cost – amounting to only 15% of the cost of large systems.

All CS-2 systems have 2 independent network layers, each a complete, independent, data network. The architecture supports up to 8. Additional layers increase bi-sectional bandwidth, reduce network contention and increase tolerance to failure.

Scalable I/O

The CS-2 architecture provides a powerful file I/O system which is both flexible and scalable. Its flexibility derives from the fact that every processing element is capable of managing its own independent I/O devices. The CS-2 operating system permits a single large file to be accessed concurrently at full bandwidth from large numbers of processors simultaneously – scalable I/O performance.

Systems are configured with a mix of devices appropriate to their I/O requirements. Each PE can be directly connected to its own disk system. In a large scientific application this enables distributed arrays to be written to local disks at very high data rates. Where concurrent I/O performance is important (e.g. in large scale database applications) each processor can control its own array of fast disks.

Network connectivity scales in the same way. Ethernet, X25, FDDI, and HiPPI interfaces can be added to as many processing elements as necessary to support the load.

Scalable Software

The CS-2 architecture provides for scalable software as well as scalable hardware. The operating system provides both administrators and users with a simple coherent view of a single system and a single hierarchical filesystem.

Applications can be written and tested on workstations and small machines prior to execution on a large production system. The same CS-2 binary will execute irrespective of the number of processing elements – applications scalability depends only upon the parallelism inherent in the algorithm.

Different applications require different types of processing. Some, such as dense matrix problems, are both highly parallel and vectorizable. Others, such as Monte Carlo simulation and Molecular Dynamics, exhibit a high degree of parallelism but are not vectorizable.

CS-2 systems are unique in offering massive parallelism with a choice of processor architecture. Premium performance is provided for vectorizable applications using a PE with supercomputer performance in a parallel or massively parallel CS-2 system. Maximum performance and workstation level cost performance is achieved on scalar applications by configuring the system with large numbers of super-scalar SPARC processors.

A system designed primarily to run one application is configured with an appropriate balance of processors for that application. A high performance compute server running a range of applications can draw on a variety of services, some scalar and some vector.

	Vector	Scalar
Processor	SPARC + μVP×2	SPARC
64-bit Speed	200 Mflops	40 Mflops
Cache/Registers	2×8 Kb	16+20 Kb
Local memory	32–128 Mb	32–512 Mb
Bandwidth	1200 Mb/sec	160 Mb/sec

CS-2 PROCESSING ELEMENTS

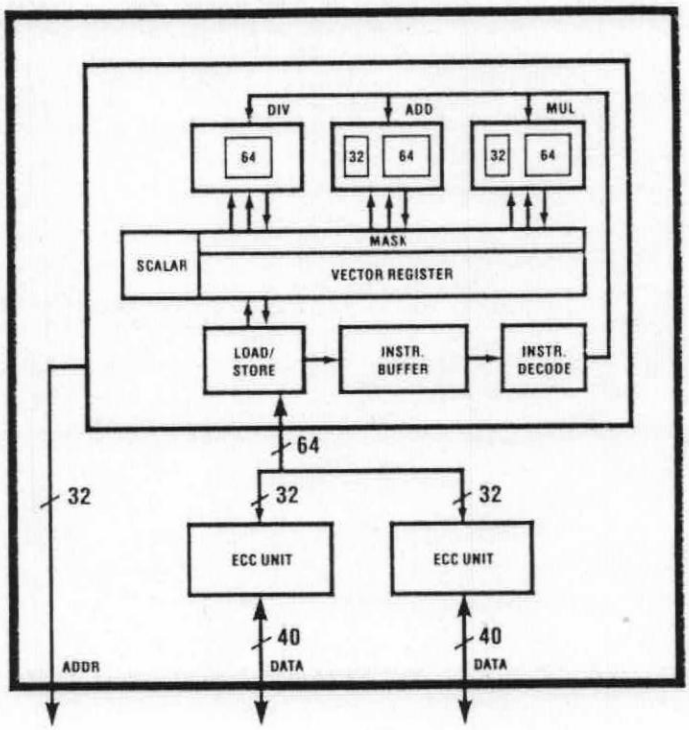
CS-2 Vector Processing Elements

Each CS-2 vector PE consists of a SPARC scalar unit, a communications processor and two Fujitsu μVP vector units sharing a three ported memory system. Cycle time is 20nS, performance peaks at 200 Mflops per PE in 64-bit arithmetic or 400 Mflops in 32-bit.

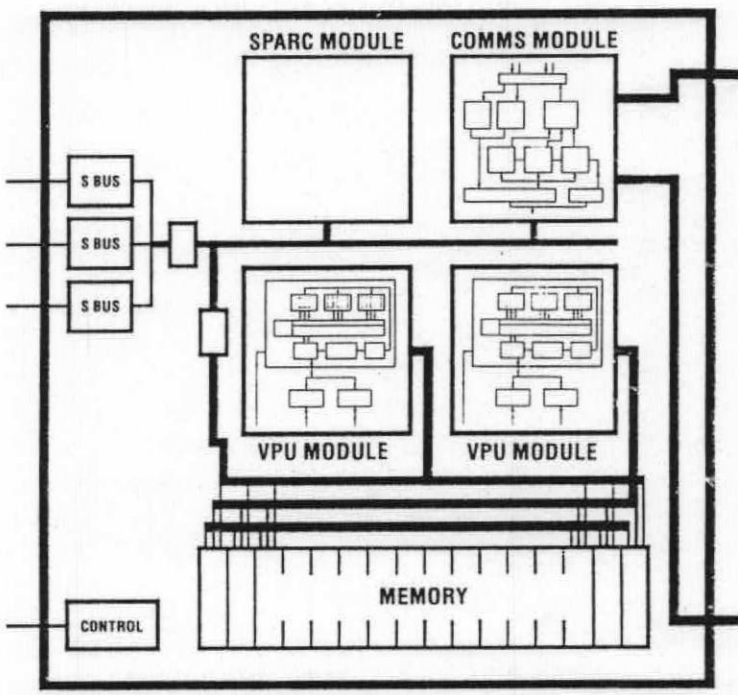
Achieving high vector performance on real-world problems requires the right balance of CPU and memory system. The CS-2 vector memory system is organized as 16 independent banks, enabling it to sustain 1.2Gb/sec on direct, strided or indirect addressing. Memory capacity is 32 or 128 Mb per PE.

The vector unit is a register to register architecture with 8Kb of flexibly configurable vector registers, 32 scalar registers, and vector mask registers whose format tracks that of the vector registers.

On-chip concurrency includes separate pipes for floating point multiply, floating point add, floating point divide, and integer operations. The floating multiply and add pipes can each deliver one 64-bit or two 32-bit IEEE format result(s) per cycle. The divide pipe



CS-2 VECTOR UNIT



CS-2 VECTOR PROCESSING ELEMENT

Scalar CPU plus 2 vector units, communications processor, 32-128Mb of memory and optional I/O interfaces.

delivers one IEEE format result every 8 cycles in either 32 or 64-bit arithmetic.

Each vector unit has its own instruction buffer and decode logic – they operate asynchronously from the scalar unit. The instruction set includes masked vector operations, compressions, vector compress under mask and expand under mask operations, as well as logical operations on integers and mask registers and conditional branches. Vector loads and stores can be performed with strides and under mask, as well as with an index vector ("indirect").

Vector register elements are scoreboarded, so that chaining between input and output operands occurs wherever possible without requiring explicit compiler or programmer intervention.

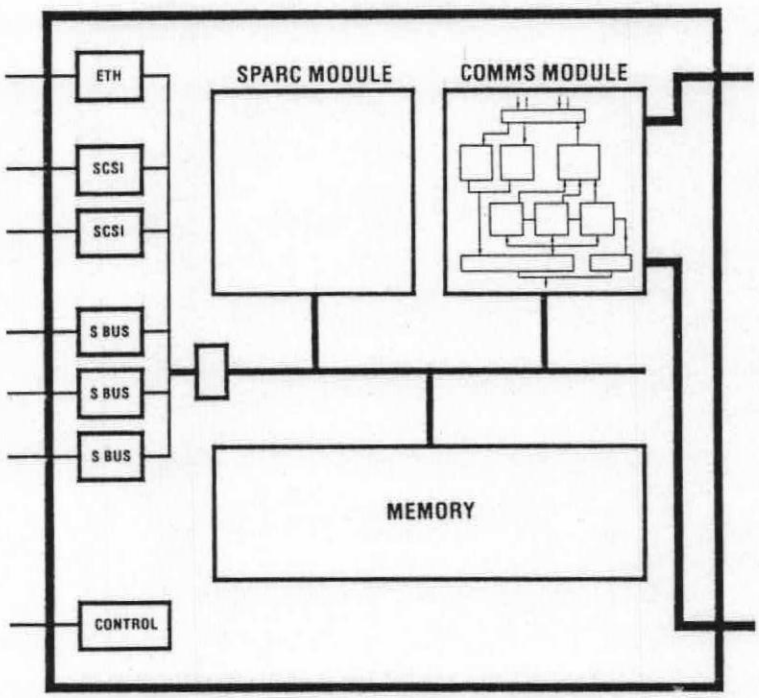
Each vector unit can issue a memory request every cycle (20nS) – a bandwidth of 400 Mb/sec and can have upto 4 requests pending. Each of the 16 memory banks can accept a new address every two cycles (40nS). In many vector and vector parallel architectures there is the possibility of contention if the vector unit generates repeated accesses to the same bank. To remove this effect CS-2 supports both the straightforward linear mapping of addresses to banks, and the option (selectable at run-time) of scrambling the allocation of addresses to memory banks. The mapping function guarantees that accesses on common strides (1,2,4,8,16,32) achieve full performance.

The CS-2 vector PE has a scalar to vector performance ratio of 1:5 – a code that is 75% vectorizable can achieve a speedup of 2.5 through vectorization. Parallel applications that are vectorizable to this extent or more will execute efficiently on a vector-parallel CS-2 system. Applications that do not exhibit this degree of vectorization may be more efficient on a system populated with scalar processing elements.

CS-2 Scalar Processing Elements

The CS-2 scalar PE comprises a super-scalar SPARC processor and a communications processor sharing a 32-512Mb memory system. The SuperSPARC CPU has two integer ALUs and an IEEE floating point unit; each is capable of generating one result per cycle. Two on-chip caches are provided: 20Kb of five-way associative instruction cache and 16Kb of four-way associative data cache. An optional second-level cache is available.

Two variants of scalar processing elements are available, one optimized for I/O intensive applications, the other for computationally intensive scalar parallel workloads. The I/O intensive variant includes an



CS-2 SCALAR PROCESSING ELEMENT
Scalar CPU, optional second level cache, communications processor, 32-512Mb of memory and optional I/O interfaces.

Ethernet interface, a pair of SCSI-2 disk controllers and three SBUS slots per PE. The computationally intensive variant is more densely packaged.

All CS-2 processing elements correct single bit and detect double bit memory errors. All memory errors are logged by the operating system.

Effective cooperation between processing elements is a crucial factor in determining the overall sustained performance of an MPP system. Maintaining effective inter-processor communication as a system scales in size is a vital aspect of preserving balance.

In designing the CS-2 architecture Meiko has concentrated on minimizing the impact of sharing work between processors. The effect of this is to increase the number of processors that can be effectively used to solve a problem, improving the performance of existing parallel programs and making parallel processing efficient for a significantly wider range of applications.

Latency and Bandwidth

In a distributed memory system, work is shared between processors by exchanging data over a communications network. The efficiency of data exchange controls the effectiveness of work sharing and hence the number of processors that can be used on a given problem. Performance is controlled by three factors:

Latency:	time spent setting up accesses to remote store
Bandwidth:	the rate at which data can be moved between processors
Concurrency:	the number of remote store accesses that can occur simultaneously.

Time spent setting up a data transfer is time spent sharing work, not time spent doing work. Time spent moving data delays its arrival and hence the time at which it can be used. Both reduce efficiency unless they can be overlapped with useful work – latency hiding.

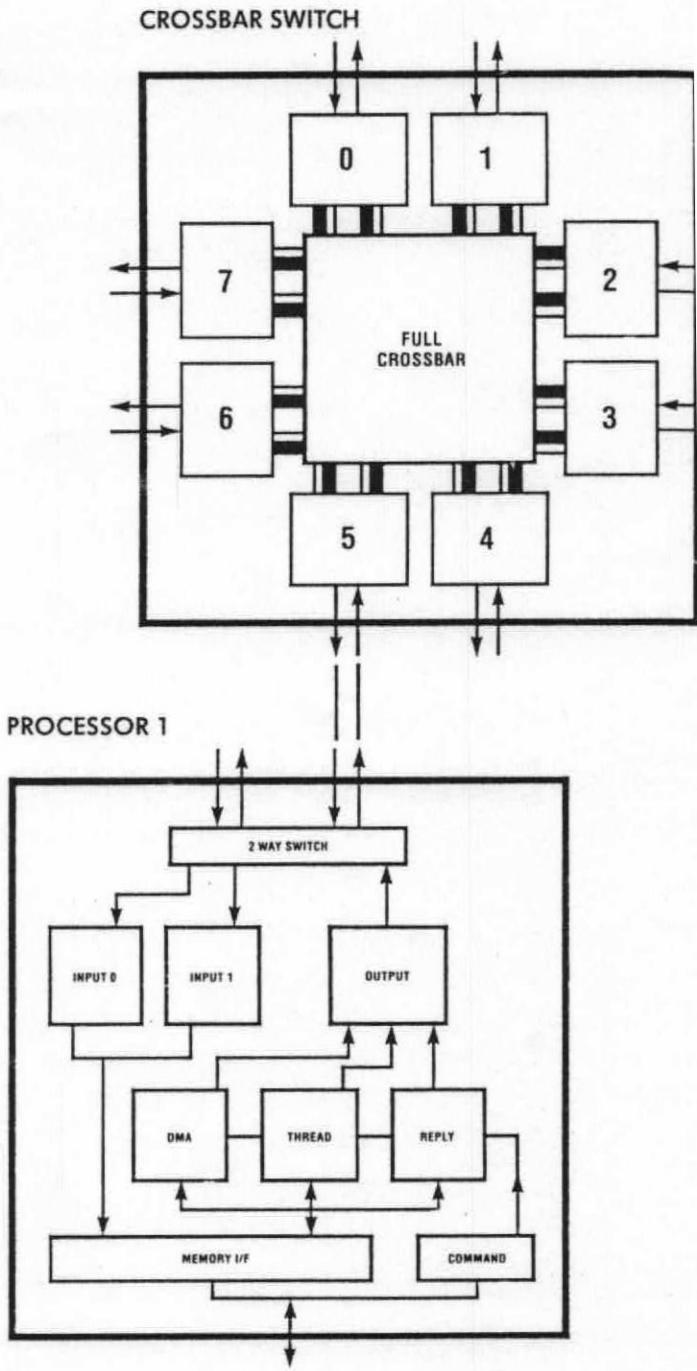
The number of concurrent data transfers depends upon the architecture of the communications network. A CS-2 data network with n PEs can sustain n simultaneous transfers between arbitrarily selected pairs of PEs at full bandwidth.

CS-2 Inter-processor Communication

Every processing element in a CS-2 system has its own, dedicated interface to the communications network: a Meiko designed communications processor. The communications processor has a SPARC shared memory interface and two data links. Data links are connected by Meiko designed 8 way cross-point switches. Each data link provides 100 Mb/sec of bi-directional user bandwidth over a physical link operating at 0.6Gbit/sec in each direction.

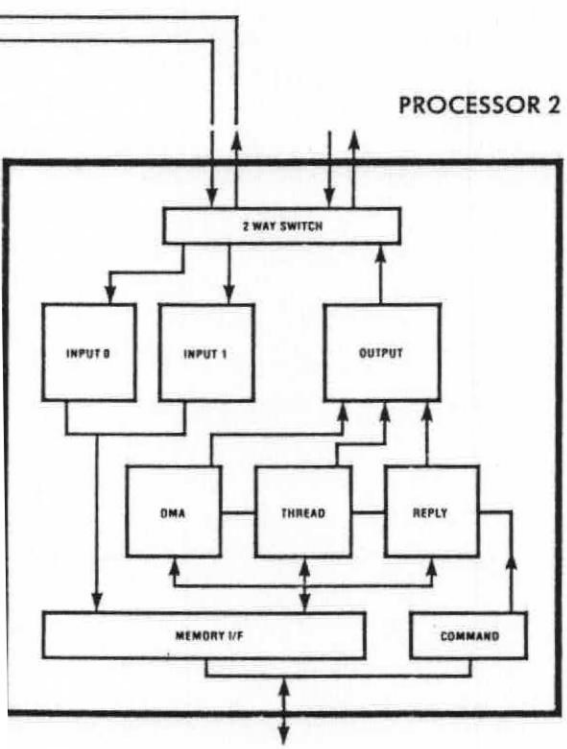
Latency is minimized in two ways. First, the communications processor manages all remote data accesses without the need for data copying, kernel intervention or main processor interrupts. Second, the use of remote store access primitives removes the synchronization overheads associated with message passing. CS-2 systems achieve remote store access latencies of 10μS at fully protected user level.

The communications processor supports remote read and remote write operations specified by virtual processor number and virtual address – both are checked in hardware. Latency hiding is supported by non-blocking instructions, instruction sequences and completion tests.



Architecture	Multi-stage switch
Link bandwidth	100Mb/sec/link
Bi-sectional bandwidth	100n/2 Mb/sec/layer
End-to-end latency	< 10μS
Network latency	< 200nS per stage
Programming model	Remote store access and synchronization

CS-2 DATA NETWORK
Each PE has its own communications processor. Communications processors are joined together by crossbar switches.



CS-2 Data Network

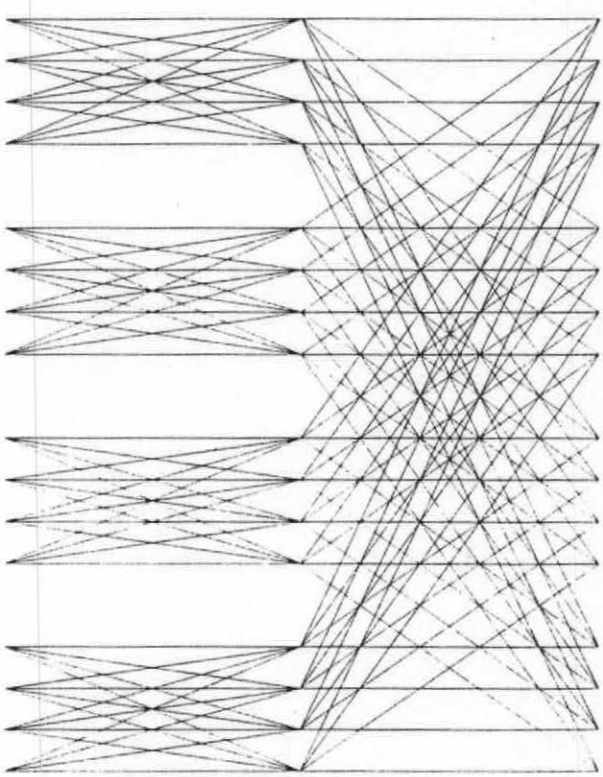
The CS-2 data network is a multi-stage packet switch, a fat tree in which the bandwidth between stages remains constant. The number of network stages is logarithmic in the number of processors: 2 stages connect 16 processing elements, 3 stages connect 64 etc. Single layer bi-sectional bandwidth grows linearly from 800Mb/sec for a 16 processor system to 12.8Gbytes/sec for a 256 processor system.

The longest path between any 2 PEs in a 256 system is through 7 network switches. This adds a maximum route delay of 1.4μS to end-to-end latencies and has no impact on bandwidth.

As well as supporting point-to-point connectivity the data network provides hardware broadcast at full bandwidth and low latency bulk synchronization.

The CS-2 architecture supports up to 8 independent layers of this switching network, 1 layer being sufficient to achieve full connectivity with multiple routes. Current CS-2 systems use 2 of the 8 layers. They are engineered to expand to use all 8 – giving a 256 element network a peak bi-sectional bandwidth of 102Gbytes/sec.

This option for performance enhancement ensures that CS-2 inter-processor communications scale in line with anticipated increases in processing power. Use of multiple layers is transparent to the application programmer.



CS-2 DATA NETWORK
64 processor, 3-stage multi-stage switch network.

CS-2 Structure

CS-2 systems are modular in construction, providing flexible configuration options and component redundancy. The basic building block is a module approximately 22×24×8 inches in size containing processor boards, switch network boards or mass storage devices. The processor module contains 4 processor boards of 1 to 4 PEs each, and the first stage of the switch network. All systems, whatever their size, are constructed from the same processor and switch network boards.

Modules are rack mounted and inter-connected in groups of 4. The 24 module system illustrated supports up to 64 vector or 256 scalar processing elements. Extension of this system is straightforward, with large systems constructed from multiple modules connected by a central switch.

Modules are individually powered and cooled. Cooling is bulk forced air with the option of internal chilled water cooling to improve thermal management of large systems.

Modules are capable of independent operation and self-test. Each contains a control system which monitors the health and performance of its processing and network elements. CS-2 supports live module insertion during operation without service interruption.

CS-2 Fault Tolerance

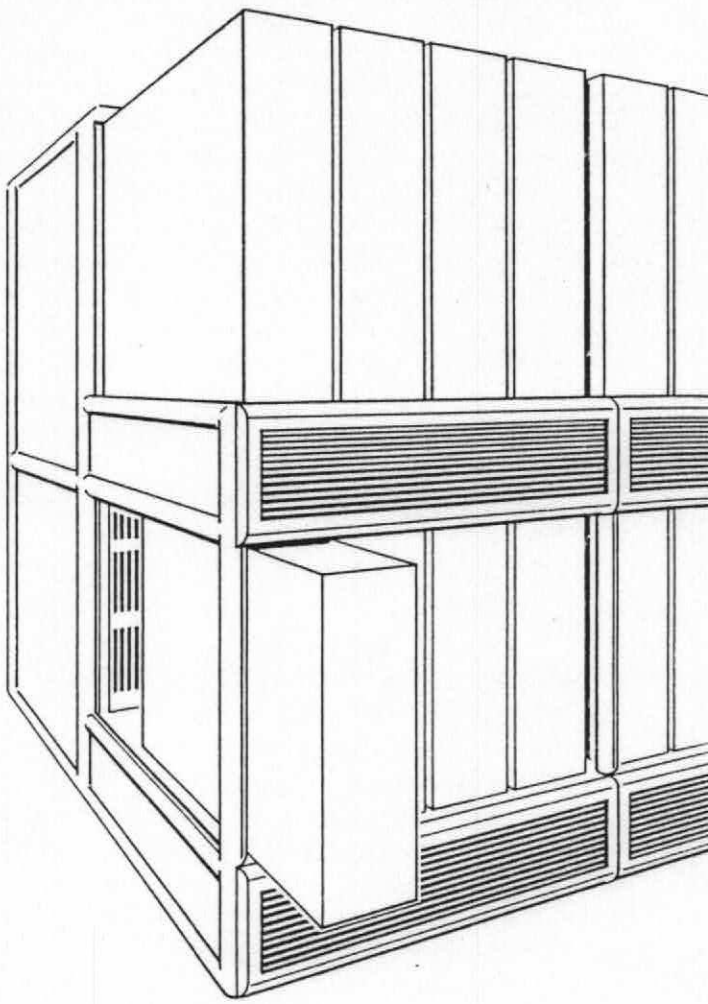
Provision of unprecedented levels of system availability is foremost in the design and implementation of the CS-2. Single points of failure have been eliminated. Errors are detected and corrected automatically where possible, detected and reported where correction is not possible.

CS-2 fault tolerance is based on guaranteeing availability in the presence of component failure. This approach extends throughout the system, from individual memory systems to whole processor modules and network layers. When combined with appropriate redundant resources the likelihood of system failure is dramatically reduced, from the probability of an error occurring, to the probability of a second error in the time taken to correct the first. Availability is increased further by the addition of multiple redundant modules.

The highest probability of failure in a large MPP system is that of soft errors in the memory system. All CS-2 memory systems use single bit error correction and double bit error detection to reduce this probability to a statistically insignificant level.

Failures in the communications network are detected in hardware in its link layer protocol using a CRC (Cyclic Redundancy Check) data integrity check. Failed network transactions are not committed to memory but generate errors on the communications processor which cause data to be resent. The network supports multiple routes between processors, allowing data to be re-routed around failed links if necessary.

The MTBF of a modern disk drive is approximately 250,000 hours, sufficiently high for a small system. However, in a large system, and when data integrity



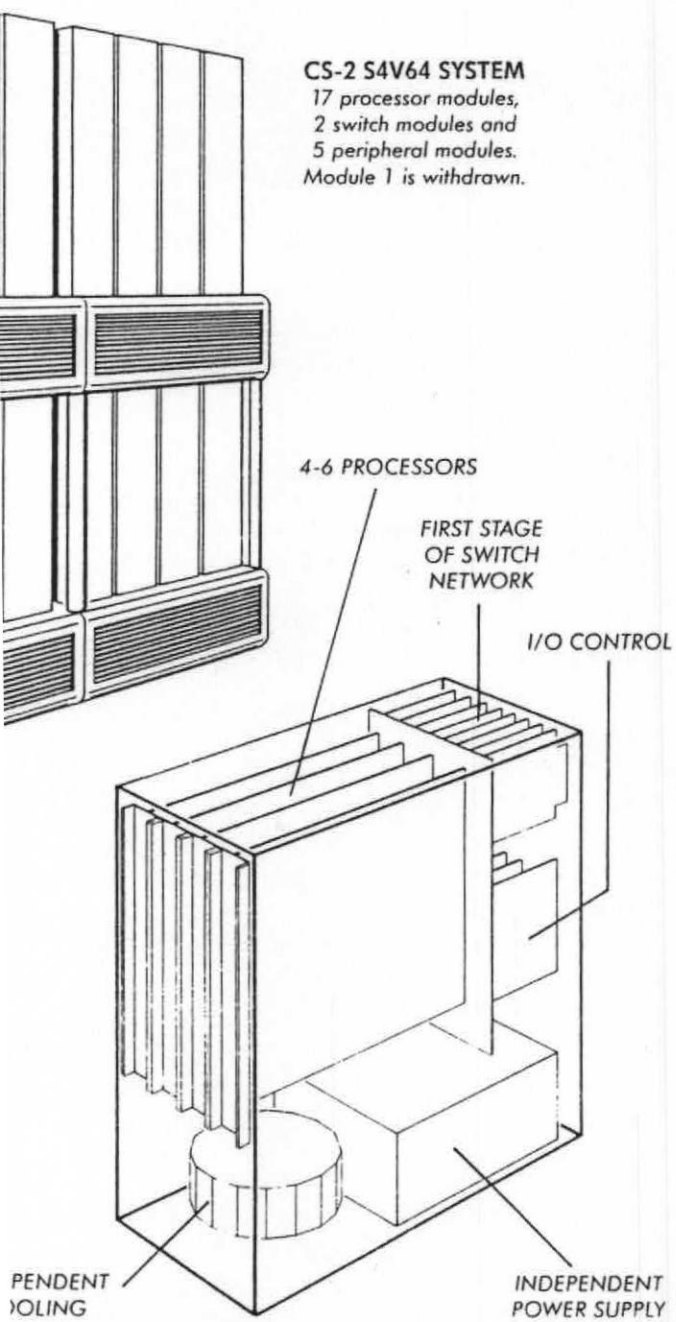
is of vital importance. CS-2 systems use dual ported RAID disk sub-systems. Each RAID sub-system of between 5 and 20 drives provides 2.4–16 Gbytes of storage capacity. Drives, controllers and power supplies are all redunded and hot pluggable in the event of failure.

The CS-2 architecture includes a control and diagnostics network which is completely independent of the data network. This network is used to monitor network performance, diagnose errors that may occur in the switch network, extract diagnostic information from individual PEs and to monitor and control power supplies and cooling systems in each module.

This network is distributed throughout the system, at board and module level. It has sufficient embedded processing power to make local decisions, issuing warnings for non-urgent classes of error and initiating module shutdown for immediate, high priority faults.

From the system administrator's point of view there maybe three types of error. Those that the system corrects itself, those that require operator intervention but do not alter the functionality of the system, and those that require replacement of a module.

This third class of serious error may cause affected jobs to be terminated, but cannot disable the operating system, which runs on at reduced capacity. In the event of such a failure a hot spare can be allocated to the domain and the job restarted. Dynamic reconfiguration does not require a system reboot. CS-2 systems can guarantee a given level of availability by redunding modules that are subject to this class of failure modes.



CS-2 S4V64 SYSTEM
17 processor modules, 2 switch modules and 5 peripheral modules. Module 1 is withdrawn.

Designed-in Expandability

CS-2 has been designed to track rapid technological development of its basic components, extending system lifetime and significantly reducing the cost of ownership.

All SPARC processors are provided on MBus modules allowing customers to upgrade PE performance while preserving investment in memory systems, infrastructure, peripherals and software. The flexibility to increase the number of processors or the power of individual processing elements permits selection of the optimal upgrade path.

A CS-2 system consists of multiple processing elements, each with a common interface to the data network. This interface is designed for longevity, allowing new and more powerful processing elements to be added to existing systems.

The inter-processor communications infrastructure has been designed with room for growth, both in terms of the link bandwidth, number of layers, and the functionality provided by the data network. Figures quoted are for current systems. This ensures that inter-processor communications performance will keep pace with increases in processing power.

Timescales for major software projects are long in comparison with the evolutionary cycle of a parallel system. Strict adherence to standard application programming interfaces combined with Meiko's commitment to high performance implementations of these interfaces ensures that applications are readily portable from one generation of technology to the next.

Meiko systems integrate smoothly into an open systems environment. They provide a reliable, high availability computational facility suitable for both interactive development and production workloads.

The CS-2 operating system is based on Solaris from SunSoft. Solaris, and conformance with the SPARC ABI, provide a stable and familiar working environment giving access to the widest possible base of UNIX applications and software development tools. Solaris conforms to the X/Open Portability Guide 3, System Five Release 4 (SVR4), and POSIX P1003.1(1990) standards.

The CS-2 operating system has been augmented in three areas :-

- Resource management
- Parallel filesystem
- Inter-processor communication

Each is vital to the performance of a massively parallel system. All standard features are identical to those of the market leading UNIX operating system. CS-2 does not require a front-end, the operating system runs on the machine.

Resource Management

The CS-2 resource management suite extends standard UNIX to support production execution of parallel applications. It includes the access control, accounting, administration, batch processing and utilization tools necessary to manage a massively parallel system.

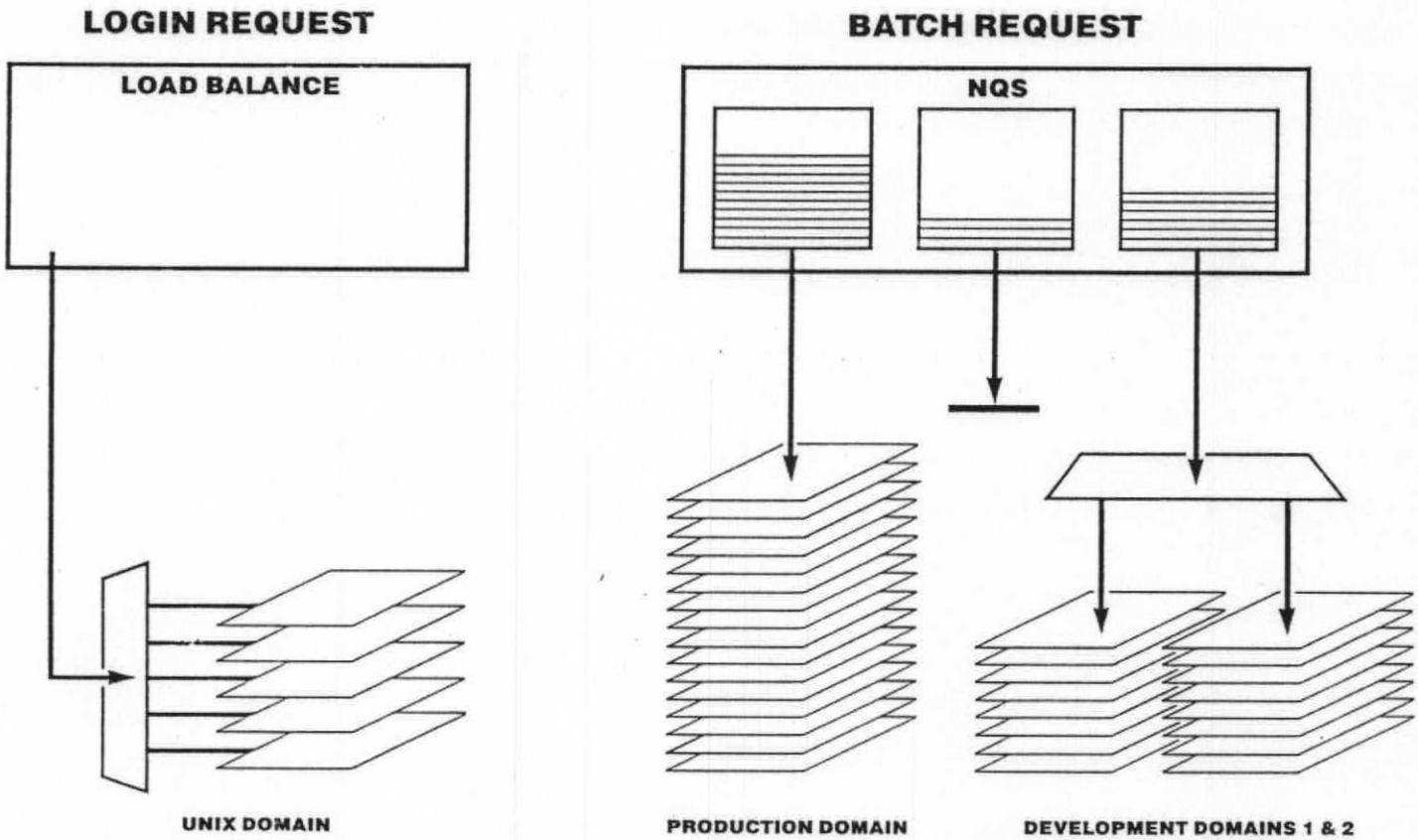
System resources, including processors, filesystems and network connections, are allocated to independently controllable groups called domains. This allocation can be changed dynamically, dedicating resources where needed. Scheduling, access control and accounting are on a per domain basis.

Users login to a domain to develop and run applications. Parallel applications are generally run on separate computation domains – a large system might have several of these. The system administrator controls user access to domains and the distribution of resources between them.

The resource manager provides full control over "administrative" parallelism in a CS-2 system – the concurrent execution of large numbers of jobs. Its GUI controls the allocation of job queues and resources to domains, as well as providing constantly updated system status and performance information.

CS-2 RESOURCE MANAGEMENT

Interactive workload and batch queues are executed by domains. PEs are allocated to domains, guaranteeing resource levels to each class of processing.



Parallel Filesystem

The parallel file system is implemented as a Solaris virtual file system which stripes the contents of its files over an arbitrary number of underlying file systems. It builds upon hardware striping used in individual devices.

These file systems may be disk or (CS-2) network based. Therefore a single file in the parallel file system may be distributed over all or any of the disks and controllers available in a system. This removes the bottleneck on seek performance and bandwidth imposed by file systems backed up by only a single disk, or a single controller, and assures scalability.

File I/O to a single processor is at data rates up to the full inter-processor communications bandwidth as data distributed over the rest of the system converges on the requesting processor. For parallel applications which may have multiple channels to disk file access rates scale accordingly.

Inter-Processor Communication

CS-2 supports inter-processor communication without the need for kernel intervention – removing the software latencies associated with remote store access. The kernel still controls permissions, and is responsible for protecting unrelated processes from each other. When processes agree to communicate through lightweight shared global objects, data is transferred without kernel intervention.

The CS-2 communications processor enforces a virtual network paradigm (analogous to conventional virtual memory) allowing the traffic of independent users and system servers to safely share the same physical hardware. This provides CS-2 with the functionality of a true multi-user supercomputer.

Software Standards

- SPARC ABI
- System Five Release 4 (SVR4)
- POSIX P1003.1(1990)
- ANSI C and Fortran-77
- Network Queuing System
- NFS, TCP/IP and OSI
- X-Windows, OpenWindows, Motif
- Application Visualization System

CS-2 Programming Environment

The CS-2 application development environment includes compilers for FORTRAN-77, ANSI C, Fortran-90 and High Performance Fortran together with a wide variety of tools for instrumenting, analyzing, debugging and parallelizing programs. This toolset runs either on the system or on networked SPARC workstations.

The FORTRAN-77 compiler conforms to ANSI X3.9-1978, with a wide range of popular extensions, including CRAY Pointers, ALLOCATABLE arrays and COMMON blocks, END DO statements, and NAMELIST I/O. The compiler also recognizes the CRAY vectorization directives.

The C compiler conforms to ANSI X3.159-1989 standards. C and Fortran are cross callable, both generate SPARC ABI compliant object code and executables.

The compilers incorporate the following standard optimizations :– constant folding, constant propagation, common subexpression removal, automatic function inlining, instruction scheduling, loop invariant removal, induction variable detection, software loop pipelining, loop splitting, loop interchange, loop vectorization, vectorization of intrinsic functions, vector idiom recognition, dead code removal, and other proprietary optimizations.

The compilation system generates code for both vector and scalar PEs. Where vector length is not known at compile time, the compiler generates both vector and scalar code: the choice of which code to execute being made at run time based on the actual vector length.

Two approaches are used for generating code for multiple vector pipes. Where there is a loop around a vector loop, the compiler will generate code which executes alternative iterations of the outer loop on each of the vector units. Where there is no outer level independent loop the compiler will allocate strips of the inner loop to each vector unit.

Parallel software is a dynamic and rapidly expanding field. Meiko is actively involved in the development of parallel programming techniques and the promotion of standard application interfaces. Two programming models are commonly used for programming MPP systems: data parallelism and multi-process parallelism.

In a data parallel application the same sequence of operations is performed in parallel on a large number of independent data items. The data parallel programming model was developed on SIMD (Single Instruction Multiple Data) machine, where the hardware constrains users to this approach. The model, however, is much more applicable, and is widely used on all types of parallel system.

In a multi-process application the problem is divided into sub-problems which are distributed over processors. This division can either be by function – different types of process handle different types of task, or by data – different processes are responsible for managing different data items. Each process operates on its own data, and accesses that of others explicitly.

The multi-process model is most powerful when an application needs to perform many different operations at the same time. Data parallelism is particularly appropriate in scientific and engineering applications dominated by repetitive operations on large arrays of data.

Both approaches are supported in full on CS-2 systems, allowing users to select the programming techniques most appropriate to their applications.

Data Parallel Programming

As an illustration of the applicability of data parallel programming consider the following Fortran-90 example

```
DOUBLE PRECISION, DIMENSION(20,20) :: x,y,z

z = a*x + y
```

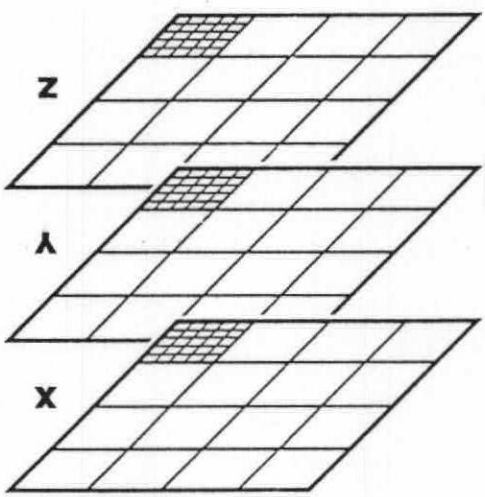
The loop can be executed concurrently for all elements of the array z. It can be run in parallel by spreading the arrays x,y and z over the available processors – each operating on a range of elements. Note that this loop is vectorizable and that if the sub-array on each processor is large enough then it can be vectorized as well as parallelized.

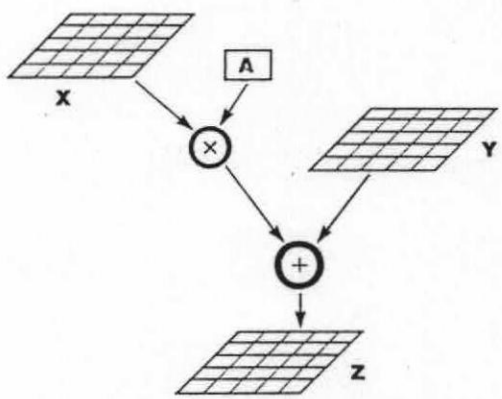
In this example all data accesses are local, no references are made to data held on other processors. When non-local data is accessed the additional latency of a remote store access is hidden. The CS-2 communications processor directly supports the asynchronous remote read and write operations needed for such non-local accesses.

A standard language for data parallel applications has been defined by the High Performance Fortran (HPF) forum – in which Meiko is an active participant. HPF is based upon Fortran-90 (which contains the standard array operations) with added data distribution statements describing the alignment of arrays against each other and the distribution of arrays of data over processors. In the HPF example below 20 by 20 arrays are aligned and distributed over a 4 by 4 array of processors.

DATA PARALLEL APPLICATIONS

perform the same operation for all elements of an array. Elements are assigned to processors. Each processor performs the same sequence of operations on each of its elements.






```
      DOUBLE PRECISION, DIMENSION(20,20) :: x,y,z
CHPF$ PROCESSORS p(4,4)
CHPF$ ALIGN WITH x :: y,z
CHPF$ DISTRIBUTE (BLOCK,BLOCK) ONTO p :: x
      z = a*x + y
```

Meiko is part of an international consortium developing an HPF compiler for the CS-2 system. This compiler builds a data parallel front-end upon the optimizations and code generation of the single processor system.

Multi-Process Programming

In the message passing model communication of data between processors is explicit. Each processor runs its own program. They can be and often are all executing the same program, but need not all be executing the same instructions at the same time. When processors need to access each others data they do so by sending and receiving messages.

The basic message passing functions are **send()** and **receive()** which move a block of data from one process to another. The sender blocks until the receiver is ready, the data is transferred and both processes continue.

The addition of non-blocking operations improves efficiency by relaxing synchronization constraints in application design. Inter-processor communication to be started as soon as possible.

On a CS-2 system the communications processor manages this inter-processor I/O while the main CPU continues to work – an example of latency hiding.

There are a wide range of interfaces to message passing. Meiko supports the standard interfaces PVM and PARMACs on CS-2, together with our own CS Tools. Intel NX/2 compatibility libraries provide portability from iPSC systems.

The CS-2 system allows synchronization constraints to by relaxed still further by providing the **Global Memory** model. A parallel application can access the memory of all of its processes without having to pass messages.

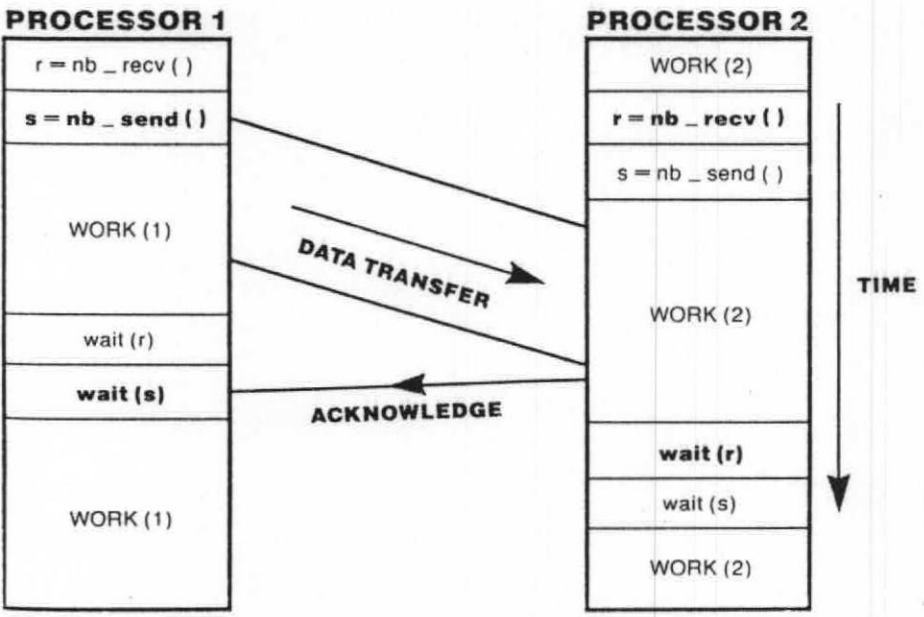
Support for global memory together with broadcast, global reduction, and barrier synchronization is provided under CSTools on CS-2.

Standard Libraries

Meiko provide a comprehensive range of maths libraries for CS-2. Optimized single processor BLAS and FFT routines are available for scalar and vector processors. Parallelized BLAS level 2 and 3 routines and multi-dimensional FFTs build on them.

Standard sequential maths libraries may by linked with parallel BLAS and signal processing libraries – perhaps the simplest way of exploiting the benefits of parallel processing.

Meiko has developed the **CS Solve** range of parallel solvers performing in memory or out-of-core QR and block-LU factorization of large dense systems of equations. CS Solve supports convergence monitoring, checkpointing and standard filesystem interfaces.



MULTI-PROCESS APPLICATIONS
can execute different operations on each processor. Work is shared by inter-processor communication. Non-blocking operations allow remote access latency to be hidden.

Parallelization Tools

CS-2 systems support a wide range of tools designed to assist in porting and parallelizing applications codes. Tools include both compiler tools for parallelizing applications and utilization tools for measuring performance. Hardware support for collecting utilization statistics is provided by the CS-2 data network.

The CS Tools multi-process debugger **pdb** provides a **dbx** style interface to debugging multi-process programs. **pdb** allows the user to set break and watch points, trace, inspect and modify variables. The GUI supports single step execution of multiple threads, each with source code listings.

The CS Tools performance monitor **csperf** provides run-time information on processor and communications network utilization. Visualization of parallel performance data is provided under AVS.

Baseline FORGE-90 includes modules for analyzing, instrumenting and maintaining large Fortran programs. Add-on modules provide both parallelization and vectorization. FORGE-90 is a highly integrated system with a user-friendly X-Windows GUI.

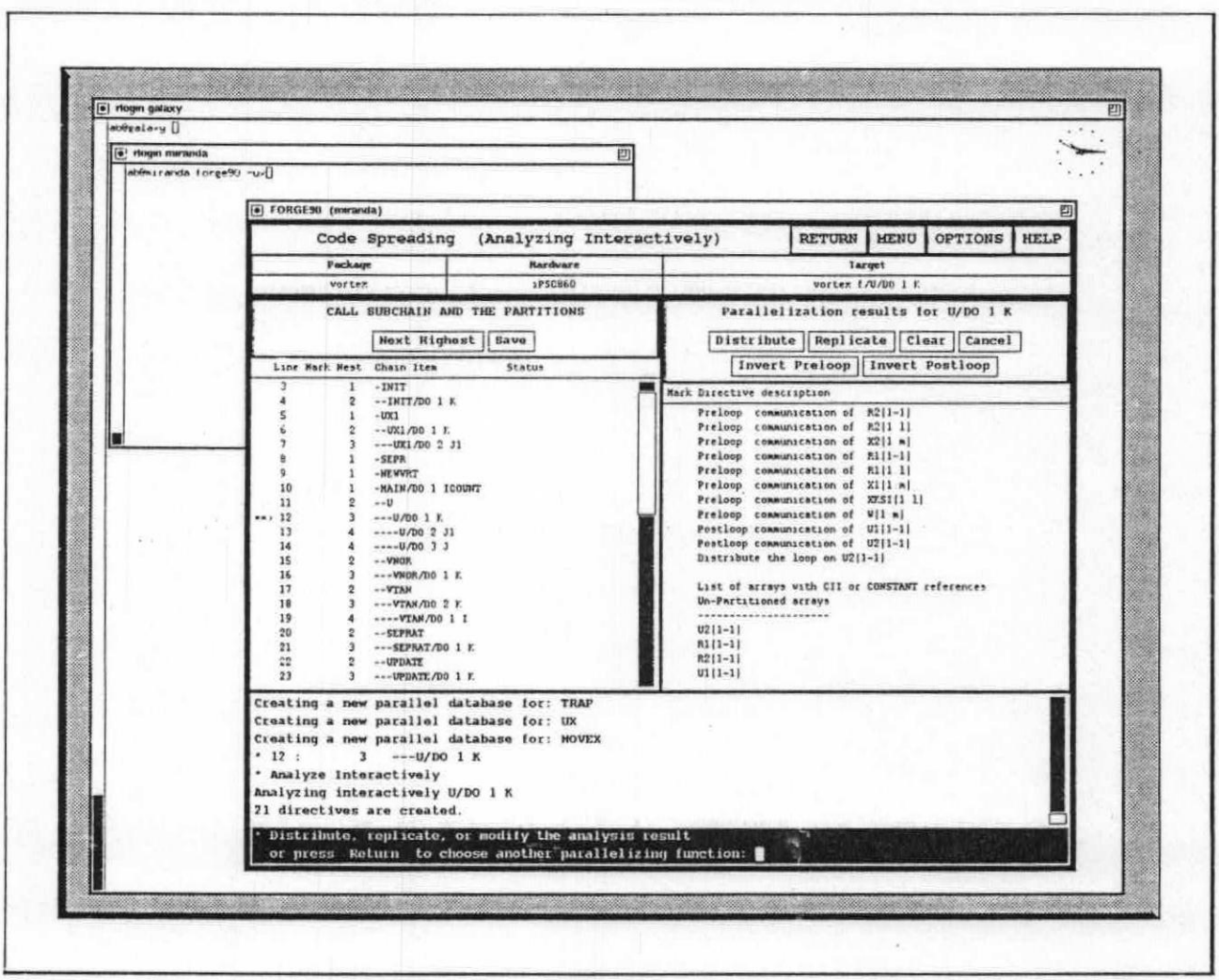
VAST-90 provides translation of FORTRAN-77 loops into Fortran-90 array operations. The Adaptor parallelization system distributes arrays over processors, automatically generating remote store access code.

Applications Packages

A wide range of packages are available in SPARC ABI format. The CS-2 architecture is designed to support this interface in full, enabling the proposed system to run all SCD 2.0 compliant software.

Meiko is involved in a program of porting key, vectorizable, packages to the CS-2 vector elements. These include the NAG, LINPACK, EISPACK and SLAP libraries, ASAS-NL, GAMESS, PAM-CRASH and PAFEC.

In addition a range of libraries including COMLIB, EISPACK, ELLPACK, LINPACK and SLAP are being parallelized for Computing Surface as are the packages AIRPLANE, AMBER, AVL-FIRE, CHARM, CTM, GAMESS, LISS, MOPAC, PAFEC, PAM-CRASH and rendering modules from AVS.



System Configuration

CS-2 systems are scalable and highly flexible. The modular architecture ensures that processing and I/O performance can be tailored precisely to individual requirements. Installations can evolve with changing needs.

CS-2 comes in 4 basic sizes, expandable to 16, 64, 256 and 1024 processors. An infrastructure upgrade is necessary to move from one to the next, but all modules are re-used.

A system may contain modules of each of the three types of processing element Vector, SPARC and SPARC plus I/O, denoted V, Q and S. Vector and SPARC plus I/O modules contain 4 processing elements, SPARC modules contain 16. Disk modules hold 2-16 Gb of data on 4-16 drives.

All systems include a standard set of peripherals:- operating system disk(s), QITC, CD-ROM, Ethernet interface, color or grayscale monitor, keyboard and mouse.

A wide range of I/O options are supported via the SPARC SBus peripheral interface. These include additional disk and tape devices, high resolution graphics displays and network interfaces for Ethernet, X25 and FDDI.

The CS-2 HiPPI interface supports peak transfer rates of up to 100Mb/sec to external framestores, disk arrays, and supercomputer networks. It includes a super-scalar SPARC processor with 64-512Mbytes of memory dedicated to protocol management.

A set of standard configurations can provide the basis for many customer systems, they range from entry level system to massively parallel supercomputers.

Model	V8	S4V16	S4V32	S4V64	S8V128	S8V256	S4Q16	S8Q64	S16Q256
Vector PEs	8	16	32	64	128	256			
Scalar PEs							16	64	256
Scalar I/O PEs		4	4	4	8	8	4	8	16
Memory (Gb)	0.5-1.5	0.5-2.5	1-6	2-10	4-20	8-36	0.6-4	2-12	8-36
Performance									
Peak 64-bit speed (Gflops)	1.6	3.2	6.4	12.8	25.6	51.2	0.8	2.9	10.9
Memory bandwidth (Gb/sec)	9.6	19.2	39	78	155	308	3.2	11.5	43.5
Network bandwidth (Gb/sec)	0.4	1.0	1.8	3.4	6.8	13.1	1.0	3.6	13.6
I/O System									
Disk capacity (Gb)	2-80	2-200	4-200	4-200	8-200	8-200	2-40	4-200	8-200
Disk bandwidth (Mb/sec)	5-40	5-100	10-200	10-200	10-200	10-200	5-20	10-40	10-80
Networking	Multiple Ethernet / FDDI / HiPPI								
Environment and Packaging									
Power Consuption (Kw)	3	10	16-20	25-30	55-60	120-150	2-5	9-12	50-60
Power	50-60Hz 220V three phase, optional uninterruptable supply								

XD126-00B100.00