New Computing Systems and their Impact on Structural Analysis and Design

Ahmed K. Noor
George Washington University, NASA Langley Research Center, Hampton, Virginia 23665, USA

INTRODUCTION

Four generations of computers are generally recognized, corresponding to a rapid change in the hardware building blocks from relays and vacuum tubes (1940s-1950s), to discrete diodes and transistors (1950s-1960s), to integrated circuits (ICs) (1960-mid 70s), to large- and very-large-scale integrated devices (mid 70s-present). These generations have increased computer performance four orders of magnitude over the last thirty years while dramatically reducing the cost (see Fig. 1). A summary of the hardware and software characteristics of the four generations is given in [1].

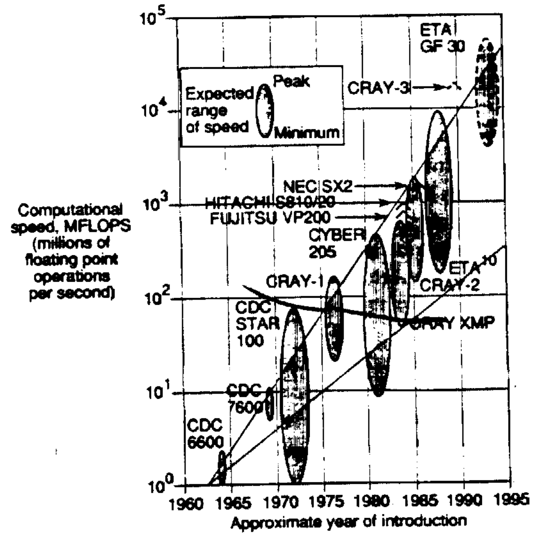

Figure 1 - Growth of computer speed.

A new (fifth) generation of computers is evolving and is likely to be available in the coming decade. The hardware building blocks for the new generation include giga-scale integrated devices [2] and [3], new transistor materials and structures, and optical components. Extensive use will be made of AI technology. Novel forms of machine architecture (e.g., new forms of parallel architecture) will be introduced and will result in a dramatic increase in computational speed. The new century will undoubtedly see more radical changes in computing technology, such as artificial neural network machines and purely optical computing. The potential of these systems is discussed in [4] and [5].

The opportunities offered by the new and projected hardware environment for structures technology are enormous. The current and evolving large supersystems, such as the CRAY-2, ETA-10, CRAY Y-MP and CRAY-3 will open the way to a vast range of new applications, and to higher levels of sophistication of current engineering problems. The small, emerging low-cost engineering workstations and personal computers will provide a high degree of interactivity and free the analysts from the constraints that are often imposed on them by large centralized computation centers. The embedded computers will aid in the control of the devices they reside in. Intelligent interfaces allowing multiple media interaction for both input and output (e.g., graphics and natural language) will facilitate the user-machine communication. Flexible high-capacity networks will allow collaborative computing by linking structural analysts and designers at different locations.

However, in order to realize the full potential of the new and emerging computing systems, the strong interrelations of numerical algorithms and software with the architecture of the systems must be understood, and special solution methodologies and computational strategies must be developed. The present paper discusses key computational requirements for future structures technology placed on evolving computer technology; summarizes some of the developments in computing systems during the recent past and near-term future; and relates these developments to structures technology.

A number of previous attempts have been made to predict the characteristics of computational mechanics software systems and the impact of advances in computing systems on structures technology and computational mechanics. The discussion presented herein is an extension and an update of those in [6] and [7].

COMPUTATIONAL NEEDS FOR FUTURE STRUCTURES TECHNOLOGY

The driving forces for future developments in the structures technology will continue to be: a) the need for improved productivity and cost-effective engineering systems; and b) support of innovative high-tech industries (aerospace related, transportation, petroleum, nuclear energy, shipbuilding, and microelectronics). In the aerospace field planned future vehicles include the national aerospace plane, the space station, improved orbital delivery systems (with large payloads, low cost and high reliability); structures subjected to very high accelerations; very high-precision shaped and controlled space structures under dynamic and thermal disturbances. The realization of the future aerospace systems requires technology advances in the structures discipline as well as in a number of other disciplines including materials, propulsion, aerodynamics, controls, avionics, optics, and acoustics. Similar advances are needed for the realization of the structures of future automotive, nuclear, and microelectronic systems. Among the technical needs for future high-performance structures are the following (see [8]):

1) Expanding the scope of engineering problems considered. This includes:

a) Examination of more complex phenomena (e.g., damage tolerance of structural components made of new material systems);

b) Study of the mechanics and failure characteristics of high-performance modern materials such as superalloys, metal-matrix composites, and high-temperature ceramic composites;

c) Study of interaction phenomena (as would be required in the thermal/control/structural coupling in space exploration, the hydrodynamic/structural coupling in deepsea mining, and the electromagnetic/thermal/structural coupling in microelectronic devices);

d) More extensive use of stochastic models to account for uncertainties associated with loads, environment, and material variability;

e) Improved representation of the details of physical systems such as damping and joints in structures (see, for example, Fig. 2); and

f) Computer simulation (modeling) of the manufacturing processes of modern materials such as solidification, interface mechanics, and superplastic forming.

2) Development of special computational strategies and effective solution methodologies for large-scale structural calculations.

3) Development of a hierarchy of models, algorithms, and procedures for the design of engineering systems. Simplified and specialized models and algorithms are appropriate for use in the preliminary and conceptual design phases and more sophisticated models are used in the detailed design phase.

4) Development of practical measures for assessing the reliability of the computational models, and estimating the errors in the predictions of the major response quantities. This also includes development of efficient adaptive improvement strategies for the computational model.

5) Continued reduction of cost and/or time for obtaining solutions to engineering design/analysis problems.

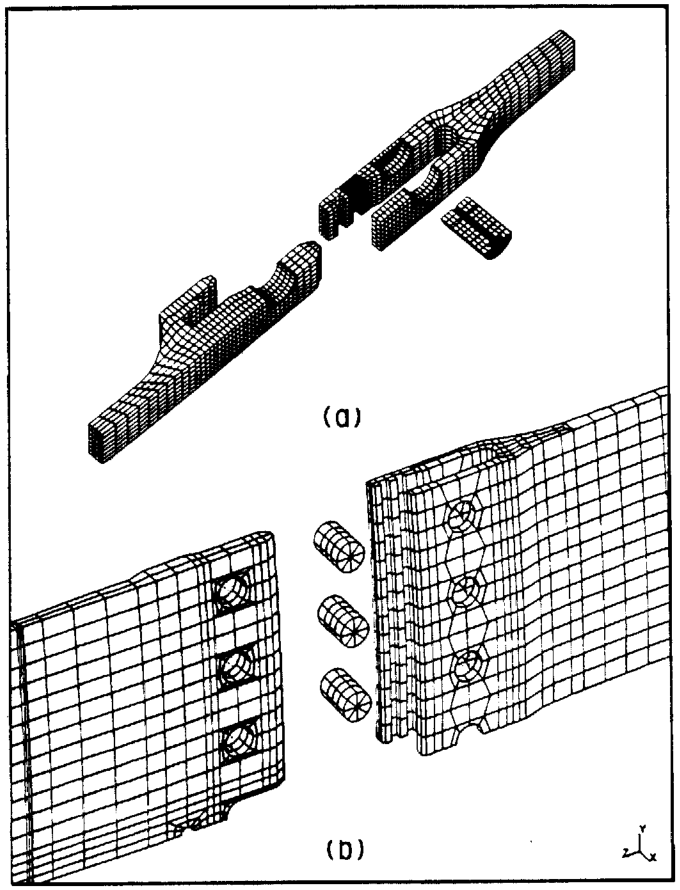

Figure 2 - Exploded views of detailed finite element models for: a) redesigned field joint of the shuttle solid rocket motor case; and b) factory joint for the shuttle motor case (courtesy of Morton Thiokol, Inc., Brigham City, Utah).

The hardware and software requirements to meet the aforementioned needs include:

1) Distributed computing environment encompassing high-performance computers (supercomputers) for large-scale structural calculations, and a wide variety of intelligent engineering workstations for interactive user interface/control and moderate-scale calculations (e.g., interactive model generation, evaluation of results in graphic form, and generation of elemental matrices). The organization of this hardware will require:

a) Extensive facilities for local and long-range networking to make all hardware and attendant software readily available to each user.

b) Development of demand-sensitive load-sharing systems to allow programs to migrate from one hardware/operating environment to another. Local networking can facilitate cooperative multidisciplinary investigations and design projects among team members interacting through shared databases. Inexpensive long-range networking can have several significant effects. Issues of proprietary data can be reduced by sending the program to the location of the owner of the proprietary data, performing the computations there, and receiving only the processed non-proprietary results.

2) Artificial intelligence-based expert systems, incorporating the experience and expertise of practitioners, to aid in the modeling of the engineering system, the adaptive refinement of the model, and the selection of the appropriate algorithm and procedure used in the solution.

3) Intelligent hardware and software interfaces allowing multiple-media interaction. These include powerful engineering workstations; high-speed long distance communication; knowledge-based management systems to incorporate verbal (audio), visual and other natural interfaces. The interfaces allow the interactive steering and dynamic control of the computations.

4) Advanced visualization capabilities, which include local high-resolution and high-speed graphics, video and film animation capabilities.

5) Integration of analysis programs into CAD/CAM systems. With the trend of moving from software-based processing to hardware-based processing, some of the analysis modules are likely to become hardware functions. Interfaces between software and hardware functions need to be designed.

6) Computerized symbolic manipulation capability to reduce the tedium of analytic calculations and increase their reliability. This allows the analytic work to be pushed further before the numerical computations start.

7) Turnkey engineering application software systems which have advanced modeling and analysis capabilities and are easy to learn and use.

BRIEF REVIEW OF CURRENT AND PROJECTED ADVANCES IN COMPUTER HARDWARE AND NETWORKING TECHNOLOGY

The major developments in computer technology have been, and continue to be, focused on improvements of cost, size, power consumption, speed, and reliability of electrical components. The next generation of computers will be impacted by the developments in three basic areas; namely, hardware components, artificial intelligence, and computer architecture and system design methods. The major advances in hardware components are briefly reviewed in this section, and some of the new computing systems are described in the succeeding sections. The survey given here is by no means complete or exhaustive; the intention is to concentrate primarily on those developments which have had, or promise to have, the greatest impact on the structures technology. Discussion is focused on microelectronics and semiconductor technology; memory systems; secondary storage devices; user-interface facilities; and networking.

Microelectronics and Semiconductor Technology

The most notable advances in hardware components in the last three decades have occurred as a result of developments in microelectronics. Instead of connecting discrete components together by wires to produce a circuit, complete circuit patterns, components, and interconnections are placed on a small chip of semiconductor material. The predominant semiconductor material in use to date is silicon. Better understanding of this material as well as better processing, tooling and packaging techniques enabled the design of fast dense circuitry. The traditional technology used for high performance logic has been *emitter-coupled logic* (ECL), which is the fastest of the silicon logic technologies, and continues to be used in most supercomputers. The *complementary metal-oxide semiconductor* (CMOS), despite its slower speed, has low power and high component density. Therefore, more gates per chip and fewer chips are used for each logic function than for ECL.

The principal advantages of microelectronic circuits are their reliability, low cost, and low power consumption. The ever-increasing number of devices packaged on a chip has given rise to the acronyms SSI, MSI, LSI, VLSI, ULSI, and GSI, which stand for small-scale, medium-scale, large-scale, very large-scale, ultra large-scale, and giga-scale integration, respectively. Since 1960 the number of components on a chip has increased continuously. For the case when no differentiation can be made between logic and memory, the progression of development is shown in Fig. 3. The density of logic chips is projected to grow at a slower rate than the density of memory chips (a factor of 7 in five years for logic chips, compared to a factor of 10 in five years for memory chips). It is anticipated that by the year 2000 the number of components per chip will reach one billion (GSI).

Developments in microelectronics were greatly aided by two projects which started in the early 1980s: the very high-speed integrated circuits (VHSIC) project supported by the U.S. Department of Defense; and the very high performance integrated circuits (VHPIC) project supported by the United Kingdom. The full range of hardware components (computer building blocks) are now available on microelectronic chips; these include memory units, addressing units (i.e., counters and decoders), complete central processing units (CPUs) called microprocessors, and even complete microcomputers (which include the CPU, memory, and input/output functions all residing on a single chip).

The net effect of the aforementioned developments has been a sustained reduction in the cost of computing.

Current research is directed towards: 1) shrinking of conductor and device

dimensions (scaling) to submicron dimensions; and 2) increasing the speed of logic circuits, to achieve a machine cycle time of the order of 1 nanosecond (1×10^{-9} second).

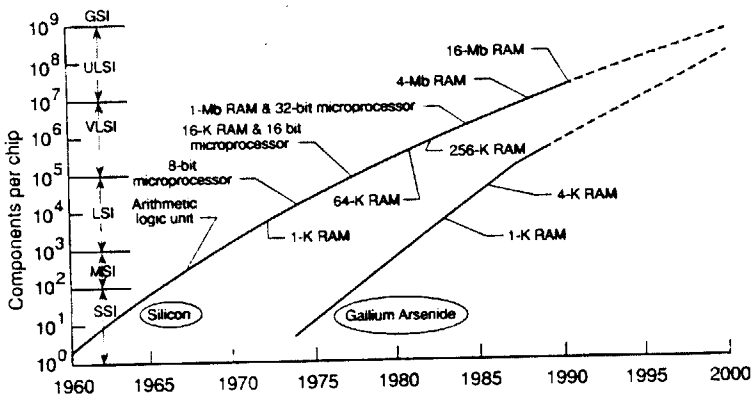

Figure 3 - Growth of number of components per ship (see [2]).

The first objective (miniaturization of electronic components) can be accomplished by using recent and improved lithography tools for etching element patterns on a chip (including optical, electron beams, ultraviolet (UV) optics, direct-wire electron beam, X-ray, and ion-beam techniques) as well as novel processing technology and fabrication techniques. Three candidate technologies are likely to achieve the second objective of ultrafast logic circuits. These technologies are based on using: 1) new material systems such as gallium arsenide (GaAs), a component semiconductor material, and super-conducting materials which do not require liquid helium temperatures (e.g., copper-oxides); 2) new transistor structures such as the quantum-coupled devices using hetero-junction-based super lattices; and 3) integrated optical circuits.

Memory Systems

Memory is the most rapidly advancing technology in microelectronics. Recent progress includes development of an entire hierarchy of addressable memories, and of high-speed, random-access memory chips with many bits of data. Each level in the hierarchy represents an order-of-magnitude decrease in access speed, and several orders-of-magnitude increase in capacity, for the same cost (see Fig. 4). The techniques of splitting and interleaving among various types of memory hierarchies in individual systems have changed some of the basic concepts of computing itself. Instead of just a few registers in the CPU and a single-level memory, a typical machine may now have:

1) a number of high-speed, general-purpose registers
2) a cache memory (or instruction buffers) for very rapid access to small amounts of data (or instructions)

3) standard central or equivalent memory
4) extended memory, directly addressable, but at a lower speed
5) hardware-implemented virtual memory, extending the amount of addressable space.

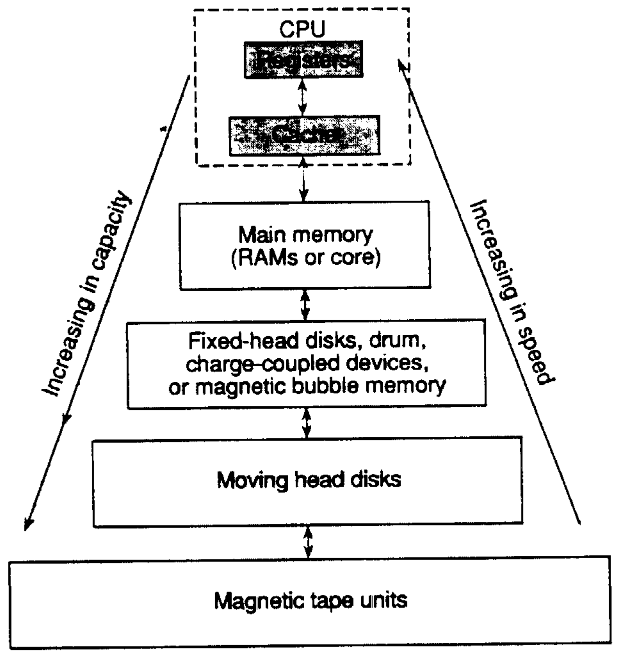
CPU
Registers
Cache
Main memory (RAMs or core)
Fixed-head disks, drum, charge-coupled devices, or magnetic bubble memory
Moving head disks
Magnetic tape units
Increasing in capacity
Increasing in speed

Figure 4 - Classical memory hierarchy (see [10]).

There are several types of semiconductor memories. These include random-access and read-only memories. In random-access memory (RAM), data can be written into, or read off, any storage location without regard to its physical location relative to other storage locations. Read-only memory (ROM) contains a permanent data pattern stored during the manufacture of the semiconductor chip in the form of transistors at each storage location that are either operable or inoperable. RAM can be either static or dynamic. Dynamic RAM (DRAM) requires constant refreshing to maintain its data, while static RAM (SRAM) does not. However, advances in DRAM permit double the amount of RAM at about the same cost as static RAM, so DRAM is used more frequently. The trends in the DRAM and SRAM chip capacity are depicted in Fig. 5. As can be seen from Fig. 5, the DRAM chips are typically one generation ahead of the SRAM chips.

Other types of memory include sequential access memories (SAMs) and direct-access storage devices (DASDs). In SAM information is accessed serially or

sequentially. Examples of SAMs are provided by shift-register memories, charged coupled devices (CCDs), and magnetic bubble memories (MBMs). DADs are rotational devices made of magnetic materials where any block of information can be accessed directly. The relative cost-access time relationship of some memory technologies is shown in Fig. 6.

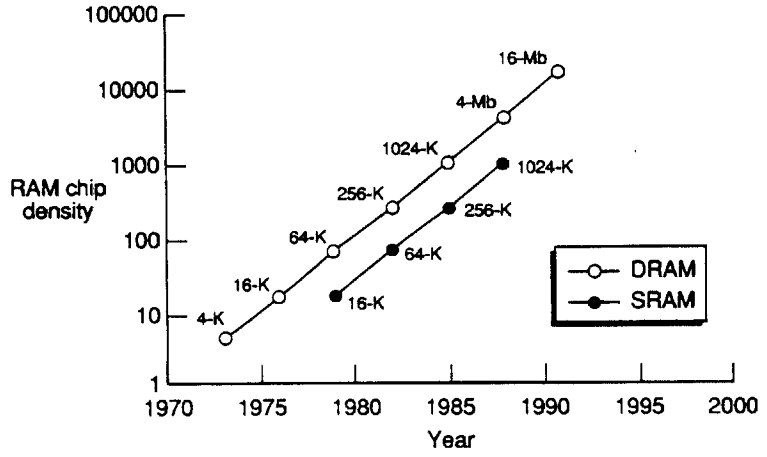

Figure 5 - Trend in RAM chip capacity.

Memory type	Typical size	Average access time
Cache	2K-128 KBytes	30-100 ns
Main memory	4K-16 MBytes	0.25-1 μs
Moving-head disk	8M-500 MBytes	25-75 ms

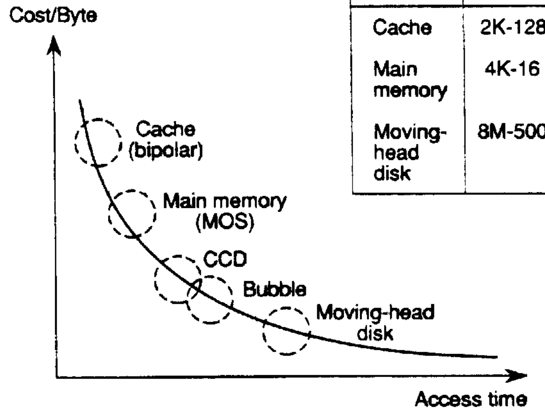

Figure 6 - Cost and access time relationship (see [10]).

Secondary Storage Devices

Magnetic devices are widely used for data storage because they offer much greater memory capacity at a lower cost per bit of data stored than semiconductor devices. As the computational speeds increase, the computers are able to utilize and produce growing amounts of data in a given period of time. This, in turn, requires an increase in the capacity of the storage devices from which (or into which) data are drawn (or loaded).

Significant improvements have been made in magnetic storage devices in the past two decades. These include the introduction of the solid-state storage devices (SSD), which are fast random-access devices used to hold pre-staged data or intermediate results which are manipulated repetitively. The SSD offer significant potential for performance improvement of more than one order of magnitude on Input/Output-bound applications, and thus allow users to develop new algorithms that would not be practical with traditional disk input/output.

The storage density of magnetic disks has increased from 200,000 bits per square inch in 1967 to its current value of over 20 million bits per square inch. Continuation of this trend is likely to yield storage densities of the order of 300 million Bytes per cubic inch within a decade. Optical storage media, such as compact disks, can provide from 5 to 7 times the density of information that magnetic devices can achieve (see [11]).

User Interface Hardware and Software

Great efforts are now aimed at improving the productivity of the analyst and designer by developing intelligent software and hardware interfaces. More structural analysis and design software are becoming *turnkey* systems with defaults built in, and with simple menu options. Current menu options are multiwindowed (one window for each task) and are controlled by lightpen, touchscreen, or mouse (which are advanced user-friendly capabilities for accessing the system). The discrete model can be generated by using either one of the geometric modeling software packages or a CAD system.

Future interface devices will support a whole range of human communication modes, and will provide the user with the freedom to choose from a variety of these modes (e.g., voice, electronic pad that responds to handwriting, sensors that track the eye movement, and a glove that enables the wearer to manipulate objects on the screen - see [12]).

Distributed Computing and Networking

High-performance engineering systems require strong collaborative analysis and design efforts, involving several engineers and machines. In support of collaborative computing, two types of communication networks are currently available: local-area networks (LAN), and wide-area networks.

Local-area networks are designed to facilitate the interconnection of a variety of computer-based equipment within a small area. They have high transmission rates

(of the order of 10 Mb/sec.), and allow different workstations to share expensive equipment and facilities. Examples of LAN are provided by Ethernet, Arcnet and token-ring network.

The first wide-area network, the ARPANET (Advanced Research Projects Area Network), was built in 1969. It demonstrated the feasibility and practicality of distributed computing, as well as of communication technology based on packet switching. A number of commercial packet networks are now available, e.g., Telenet and TYMNET. Moreover, the coupling of digital networking with the existing telephone and Digital Private Branch Exchange (PBX) systems into Integrated-Services Digital Networks (ISDNs) promises to offer access to a wide range of data and central computers via desktop workstations.

Current work is directed towards increasing the transmission rates of both local-area and wide-area networks (see [13]). Future directions include development of networking technology to support portable computing and communications. The cellular phone is likely to evolve into a portable voice and data machine that supports mobile communications.

CLASSIFICATIONS AND PERFORMANCE EVALUATION OF NEW COMPUTING SYSTEMS

Because of the rapid progress made in recent years in component technology, a number of novel forms of computer architectures have emerged. Some of the new architectures are commercially available; others are still research tools aimed at achieving high-performance and/or low-cost computations. In this section the classifications and performance evaluation of different machines is discussed, and in the succeeding sections, the major features of the new and emerging high-performance machines are described.

Classifications of Computing Systems

In an effort to identify and clarify the differences between the different machines, a number of classifications and taxonomies have been proposed. One of the earliest and most commonly-used classifications is that introduced by Flynn [14], which is based on the concurrency in instruction control and concurrency in execution. A stream is defined as a sequence of items (instructions or data) as executed or operated on by a processor.

Four broad classes can be identified according to whether the instruction or data streams are single or multiple (see Fig. 7).

1. *Single-Instruction-Stream, Single-Data-Stream (SISD) Machines.* These machines include the conventional serial computers which execute instructions sequentially, one at a time.

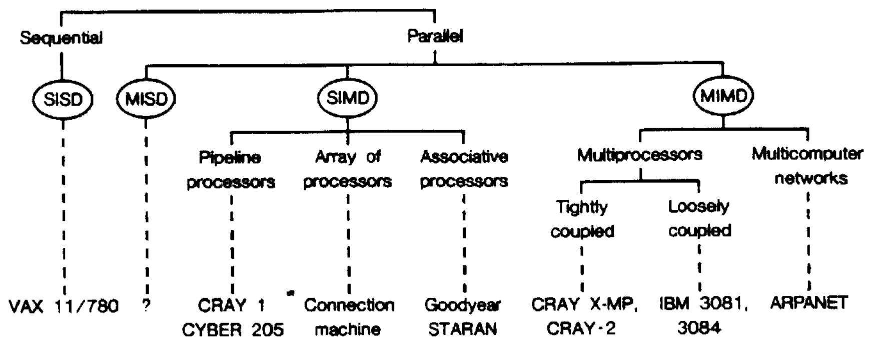

Figure 7 - Classification of computing systems

2. *Single-Instruction-Stream, Multiple-Data-Stream (SIMD) Machines.* These are computers that have a single control unit, a collection of identical processors (or processing elements), a memory or memories, and an interconnection network which allows processors to exchange data. During execution of a program the control unit fetches and decodes the instructions and then broadcasts control to the processing elements. Each processor performs the same instruction sequences, but uses different data. These operations are usually referred to as lockstep operations.

3. *Multiple-Instruction-Stream, Single-Data-Stream (MISD) Machines.* This category is included to round out the four possible combinations of single/multiple instruction and single/multiple data, but none of the current machines is classified as MISD.

4. *Multiple-Instruction-Stream, Multiple-Data-Stream (MIMD) Machines.* These are computers which contain a number of interconnected processors, each of which is programmable and can execute its own instructions. The instructions for each processor can be the same or different. The processors operate on a shared memory (or memories), generally in an asynchronous manner.

MIMD machines can generally be divided, according to the level of interaction between processors and their physical location, into multiprocessors and multicomputer networks. The latter class refers to physically dispersed and loosely-coupled computer networks. Multiprocessors can be further subdivided into tightly-coupled and loosely-coupled systems (see Fig. 6). In a tightly-coupled system the processors access a global, shared memory by means of some interconnection networks. In a loosely-coupled system, each processor has its own local memory and communicates with other processors by sending messages through the interconnection network.

Examples of computing systems which belong to each of the classes are given in Fig. 7. Also, block diagrams of the SISD, SIMD and MIMD machines are shown

in Fig. 8. The architectural evolution of computing systems from sequential scalar processors to concurrent vector/scalar processors is depicted in Fig. 9. The trend has been to build more hardware and software functions into the system. The skewed tree demonstrates that most current high-performance computers are designed with look-ahead techniques, functional parallelism, pipelining at various levels, using explicit vectors, and exploiting parallel processors in SIMD or MIMD mode. Note that some of the SISD machines have parallel processing mechanisms (due to the presence of multiple functional units and/or facilities for overlapping computations and I/O); however, the parallel processing is embedded in the hardware below the instruction level, and the appearance of sequential execution of instructions is preserved.

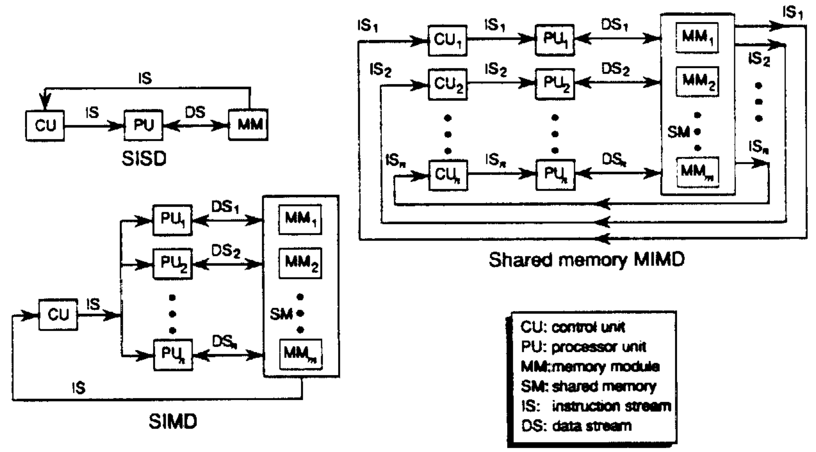

Figure 8 - Block diagrams for SISD, SIMD and MIMD machines

Performance Evaluation of Computing Systems

1) *Peak Versus Sustained Performance of Machines.* The performance of sequential computers is usually measured by a) the peak computational speed measured in millions of floating-point operations per second (MFLOPS), and b) the peak execution rate of arithmetic, logic, and program control instructions, measured in millions of instructions per second (MIPS). In the case of SIMD and MIMD machines, the *peak computational speeds* in terms of millions of floating-point operations per second can be estimated. However, the *sustained computational performance* in these machines is difficult to estimate since it varies with the level of parallelism achieved, and the overhead incurred in exploiting the parallelism in the particular application [16]. These, in turn, are functions of the formulation used, the numerical algorithm selected, the computer implementation, the compiler and the operating system used, as well as the architecture of the hardware. For vector multiprocessors the peak performance can be estimated as the sum of the vector performance of all the processors, and the lowest performance is that of a single-

scalar processor. A widely used approach for estimating the sustained performance of a machine is *benchmarking* - running a set of well-known programs on the machine. A discussion of the effectiveness and pitfalls of benchmarking is given in [17].

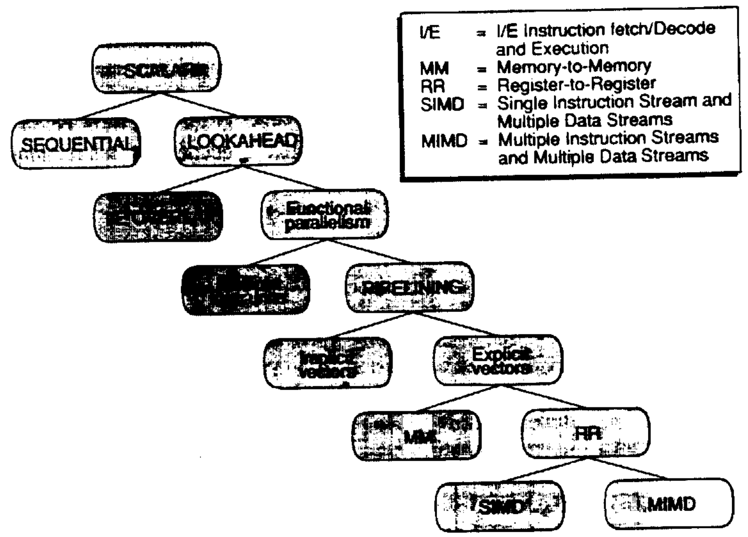
SCALAR
SEQUENTIAL
LOOKAHEAD
Functional parallelism
PIPELINING
Implicit vectors
Explicit vectors
MM
RR
SIMD
MIMD
I/E = I/E Instruction fetch/Decode and Execution
MM = Memory-to-Memory
RR = Register-to-Register
SIMD = Single Instruction Stream and Multiple Data Streams
MIMD = Multiple Instruction Streams and Multiple Data Streams

Figure 9 - Architectural evolution of computing systems (see [15])

2) *Amdahl's Law.* In 1967, Amdahl made the observation which has come to be widely known as Amdahl's Law [18]: If a computer has two modes of operation - one high speed and the other low speed - then the overall performance will be dominated by the low-speed mode. Amdahl's Law is fundamental to the understanding of computer performance. It shows that a bottleneck in a computing system is associated with a small execution rate (low-speed mode of operation) which is out of balance with the rest of the execution rates (high-speed modes of operation). In such a computing system increasing the execution rates for the high-speed modes of operation may yield only a small increase in performance unless the fraction of computation performed in the low-speed mode is essentially zero. This explains why vector computers with low-scalar speeds have not been successful. A recent discussion of the application of Amdahl's Law to massively parallel processors is given in [19].

MAJOR FEATURES OF NEW COMPUTING SYSTEMS

New computing systems cover a broad spectrum of machines ranging from the large supersystems to the small portable, embedded computers, and transputers. A description of some of these machines is given in [15, 20 and 21]. Herein, a brief discussion is given of the major features of new computing systems that are likely to have the strongest impact on the structures technology.

Most of the new and emerging machines achieve high performance through concurrent activities in the computer (or the network of computers). The exploitation of these concurrent events in the computing process is usually referred to as *parallel processing*. When parallel processing is done on physically dispersed and loosely coupled computer networks, it is usually referred to as *distributed processing*. The concurrency is used not to speed up the execution of individual jobs, but to increase the global throughput of the whole system.

Parallel processing can be applied at four distinct levels, namely: job level, program level, inter-instruction level, and intra-instruction level [10 and 22]. The hardware and software means to achieve parallelism in each case are outlined in Table 1. The hardware role increases as the parallel processing goes from high (job) level to low (intra-instruction) level. On the other hand, the role of software implementations increases from low to high levels.

Supersystems

Supersystems are a class of general-purpose computers designed for extremely high-performance throughput. Although there is no universally accepted definition or classification of supersystems, the following three classes of supersystems can be identified (based on the performance, memory system used, and cost):

- large supersystems
- near supers
- minisupers

The current price ranges for the three classes in U.S. dollars are: $2M to $20M; $1M to $4M; and $100K to $1.5M, respectively.

Current *large supersystems* have the following four major characteristics: 1) high computational speeds (maximum speeds of the order of 200 MFLOPS or more); 2) high execution rates (of the order of 500 MIPS or more); 3) large main (or central) memory (with a capacity of 8 MBytes or more); and 4) fast and large secondary memory (or storage devices) with a sophisticated memory management system.

The development of large supersystems now spans three generations. The first generation included the array of processors ILLIAC IV (SIMD machine); and the pipeline (or vector) computers CDC STAR-100, and Texas Instruments Advanced Scientific Computer (ASC). The second generation supersystems included the CRAY-1, which featured the pipelined vector instructions introduced in the first generation machines, but carried out in a clever register-to-register mode. They also included the CDC Cyber 203 (an enhanced successor of STAR-100, sporting a faster scalar unit). Most of the current third-generation large supersystems use a hybrid combination of pipeline and array processors to achieve high performance. Examples of these machines are: the CDC CYBER 205, the CRAY X-MP, CRAY-2, CRAY Y-MP, ETA-10 and the Japanese machines Fujitsu VP-400, Hitachi S810/20 and S820/80; and NEC SX2-400. The top sustained speed of current supersystems is of the order of 500 MFLOPS with bursts to 1.5 GFLOPS (billions of floating point operations per second).

Table 1 - Levels of parallelism

Level	Means to achieve parallelism	Performed by
Job level	**Multiprogramming** - overlapping and interleaving the execution of more than one program (I/O and CP operations) **Multiprocessing** - running two or more CPU's concurrently on different applications or on independent job streams of the same general application	Operating system
Program level	**Multitasking** - decomposition of a program into two or more tasks (program segments) that can execute concurrently. This requires the tasks to have no data or control dependence. **Microtasking** - which permits more than one CPU to work on a program at the Do-loop level	Software
Inter-instruction level	**Concurrency among multiple instructions.** This requires an analysis of the data dependency and is accomplished by dividing each instruction into suboperations, and overlapping the different suboperations on different instructions.	Compiler
Intra-instruc-level	**Pipelining** - by dividing the instruction into a sequence of operations, each of which can be executed by a specialized hardware stage that operates concurrently with other stages in the pipeline. **Very-long instruction word (VLIW)** - performing multiple operations per instruction, each with its own address field.	Hardware

The next generation of supersystems includes the CRAY-3 and CRAY-4, the Japanese Super Speed Computer project; the Japanese fifth generation project; and the IBM TF-1 Project (an SIMD machine, with 32,768 processor nodes, capable of executing 10^{12} flops). It is anticipated that the computational speeds of the supersystems will continue to increase, and will exceed 20 GFLOPS in the next five years. The architectural characteristics of some of the new and emerging large supersystems are given in Table 2.

Near supers are high-end main frames with peak computational speeds in the range of 50-500 MFLOPS; execution rates of the order of 50 MIPS, or more; and less sophisticated memory systems than those of the large supersystems. Examples of these machines are IBM 3090/VF, CDC Cyberplus, the Connection Machine C-1 of Thinking Machine, Inc., and the BBN Butterfly.

The *minisupers* are low-cost parallel/vector computers with peak computational speeds in the range of 10 to 100 MFLOPS, and execution rates of the order of 10 MIPS, or more. Examples of these machines are the Encore/Multimax, Convex C-1 and C-2, Intel iPSC system, FPS models 300, 350 and 500 of FPS Computing (formerly Floating Point Systems, Inc.).

Supersystems can impact the structures technology in a number of ways including:

a) Increasing the level of sophistication in modeling structures to new levels which were not possible before. Examples are provided by reliability-based (stochastic) modeling of structures (to account for probabilistic aspects of geometry, boundary conditions, material properties and loading), and multidisciplinary analysis and design of structures.

b) Reducing the dependence on extensive and expensive structural and dynamic testing. This is particularly important for future large space structures (e.g., large antennas, large solar arrays, and the space station) where the reliability of testing in 1-G environment can be, at best, questionable; and,

c) Enhancing the physical understanding of some aspects of mechanical behavior which are difficult, if not impossible, to obtain by alternate approaches. An example of this is the study of the implications of various microstructural mechanisms of inelastic deformations on the macroscopic response.

Parallel Processing Machines
In the last few years there has been an explosion in the number of parallel processing machines developed. Some of these machines belong to the class of supersystems discussed in the preceding subsection. In an effort to identify and clarify the differences between these machines, a number of classifications and taxonomies have been proposed, including Flynn's classification described in the preceding subsection. However, Flynn's classification does not adequately describe several of the new multiprocessor machine architectures. More descriptive classifications of multi-

Table 2 - Architectural characteristics of some of the new and emerging large supersystems

System/ Model	Architecture/ Configuration	Maximum Number of Processors	Processor Type	Clock Cycle (ns)	Maximum Main-Memory Capacity	Peak Computational Rate (GFLOPS)
CRAY X-MP	Multiprocessor	4	Custom	8.5	64 MW	0.84
CRAY-2	with	4	ECL	4.1	256 MW	1.72
CRAY Y-MP	shared	8		6.0	32→128 MW	2.7
CRAY-3	memory	16	Ga As/ECL	2.0	128→512 MW	16
ETA-10G	Multiprocessor with local and shared memory	8	Custom CMOS	7.0	16 MW/processor	10
IBM ES3090-600S	Multiprocessor with shared memory	6	ECL	15.0	512 MB	0.8
Fujitsu VP 400E	Single processor with multiple pipelines	1	ECL	14.0-scalar 7.0-vector	1GB	1.71
Hitachi S-810/20		1	Custom	14.0	256 MB	0.8
Hitachi S-820/80		1		4.0	512 MB	3
NEC SX2-400		1	MOS	6.0	1 GB	1.3
NEC SXX	Multiple processor	4		3.0		20

Notes: 1 MB (MW)=1,048,576 Bytes (words-each word 64-bits); 1 GB(GW)=1,073,741,824 Bytes (words-each word 64 bits); 1 GFLOP = 1 billion floating point operations per second; ECL = emitter-coupled logic; MOS = metal-oxide semiconductor, CMOS=complementary metal-oxide semiconductor.

processor systems are based on the following characteristics (see [23]): processor granularity; memory organization, connection topology between processors and memory systems; reconfigurability (to meet the performance needs in more than one environment); control mode (command flow versus data flow); extensibility (e.g., possibility of adding more processors without pronounced decrease in the efficiency of the system) and homogeneity or nonhomogeneity of the processors (for example, the CRAY X-MP has homogeneous processors, but a combination of a serial computer and an attached processor can be viewed as a heterogeneous multiprocessor). Herein, an extension of Flynn's taxonomy, proposed in [22] is described. Also, the classifications based on the first three characteristics, namely, processor granularity and speed; memory organization; and connection topology, are discussed (see [22]).

Extension of Flynn's Taxonomy. The extended taxonomy uses levels of concurrency as one principle, and instruction types as another. Four generic types of instructions and three levels of control concurrency are included in the taxonomy. The four generic types of instruction are: *scalar*, *vector*, *systolic* and *very-long instruction word* (VLIW). They are distinguished by the number of operations and number of pairs of operands as shown in Table 3.

The three levels of control concurrency include the *serial* and *parallel* types (with single and multiple instruction streams), as well as the *clustered* type (with several independent sets of multiple instruction streams). The three levels constitute a hierarchy, with parallel control being a generalization of serial control, and clustered control being a generalization of parallel control.

The extended taxonomy has twelve types of computer architecture. Examples of some of these types are given in Table 4.

Processor Granularity and Speed. Multiprocessor systems can be divided into three groups:

a) *Coarse-grain machines* consisting of a small number of very powerful processors or central processing units (CPUs) linked together. Examples of these machines are CRAY-2, CRAY Y-MP and ETA-10.

b) *Fine-grain machines.* These are massively parallel machines which combine thousands of relatively weak processors. When the processors work in concert they can form very powerful computers. Examples of machines with synchronous processors are the ICL Distributed Array Processor (DAP), with 1024 processors, at Queen Mary College in London; the Goodyear Aerospace Massively Parallel Processor (MMP), with 16,384 processors; the Connection Machine C-2 of Thinking Machine, Inc. with up to 65,536 single-bit processors; and the FPS Computing T-Series hypercube with up to 16,384 processors.

c) *Medium-grain machines.* These fall somewhere between the first two categories. They have a moderate number (e.g., tens or hundreds) of low-cost and mid-range processors (performance of the order of 2 MFLOPS for the former and

10-100 MFLOPS for the latter). Examples of the former are the Encore Multimax, Flexible Computer's FLEX/32, Sequent Balance 8000, and the Intel iPSC system. Examples of the latter are the Alliant FX/8 and Elxsi 6400 systems.

Table 3 - Classification of instruction types (see [23]).

Type of Instruction	Number of Operations	Number of Pairs of Operands	Examples
Scalar	1	1	A = B + C
Vector	1	N	$A_i = B_i + C_i$, i = 1,N
Systolic	M	1	Matrix operations, with data from rows and columns used repeatedly.
VLIW	M	N	Multiple operations per instruction, each with its own address field

Note: N, M > 1

Table 4 - Extended taxonomy of computer architectures (see [23]).

Level of Concurrency	Generic Instruction Types			
	SCALAR	VECTOR	SYSTOLIC	VLIW
SERIAL	CDC-7600 IBM 360/95	CRAY-1 CYBER 205	WARP	
PARALLEL	BBN-Butterfly Hypercube	CRAY Y-MP ETA^{10}		
CLUSTERED	Myrias	Cedar		

Memory Organization. One of the most important classifications of multiprocessor systems is that based on memory organization. According to this classification there are shared-memory machines, private- (or semiprivate) memory machines, and multilevel-memory machines.

Shared (or global) memory. This is a single monolothic main memory accessible to all processors. Examples are provided by the memories of the CRAY X-MP, the Ultracomputer of New York University, and the IBM ES3090-600S.

Distributed memory. This is a private memory connected to a single processor and requiring communication to transfer information. Examples are provided by the memories of the Connection Machine, the FPS T-Series, and the Intel iPSC.

Semidistributed memory is one which is available to a subset of processors.

Multilevel (hierarchical) memory is available in computer systems such as the ETA-10 and the Myrias 4000.

Connection Topology. Multiprocessor systems with private (or semiprivate) memories can be distinguished by the interconnection pattern (or topology) between processors, memory systems and I/O facilities. The topology affects the class of problems which can be efficiently solved on the machine. The commonly-used connection topologies are:

Bus (or ring) type connection. The various processors, memory systems, and I/O facilities reside on a common communication bus or set of buses. Most computers of this type incorporate global (or central) memory, shared by all processors, and accessed via the communication bus. Examples of machines with bus connection are the Encore Multimax, FLEX/32, and Elxsi 6400 computers.

Hypercube or n-cube connection. Each processor is directly connected to a number of its neighbors, n. The number of connections per processor results in a multidimensional cube with 2-inch nodes. This type of connection is usually used with distributed memory machines. Examples of machines with hypercube connection are the Intel iPSC and the FPS T-Series.

Switching connection. This mechanism is based on placing a switch between the processors and memory banks, thereby removing the ownership property between processors and memory. For different settings of the switch, a given processor will be connected to different memory banks.

Although most of the currently available machines for performing parallel computations belong to either SIMD or MIMD categories of Flynn's classification, there are a number of variations. Two of these variations which exploit parallelism in ways which have promise for achieving high performance are *data-flow machines* and *systolic array architectures.*

The ***data-flow concept*** is based on the dependency graph of a computation. The algorithm for a given computation is first written in a special programming language designed for data-flow applications (e.g., the language Id developed at the Massachusetts Institute of Technology). A compiler then translates the program into a data flow graph (which corresponds to the machine language for data flow architecture).

Systolic architectures follow from space and time representations of certain numerical algorithms which map directly onto geometrically regular VLSI/WSI (Very Large-Scale Integration/Wafer-Scale Integration) structures. The term systolic array comes from the notion of data pulsing through the processors in the network in an analogous manner to that of blood pulsing through the circulatory system in the body (see [24]). In its purest form a systolic system consists of a regular array of processing elements all doing the same calculation and passing results on to their nearest neighbors every cycle. In this way the array as a whole computes some recurrence function. The prime features which make this style attractive are the ***short interconnections***, the ***regularity*** which gives a high packing density and simplifies the design, and the high degree of ***parallelism*** which, when combined with the other features, leads to high performance circuits. Potential applications of systolic arrays in finite element applications are discussed in [25, 26 and 27]. More advanced concepts of systolic arrays and systolic computing are described in [28].

Parallel processing systems can substantially expedite the multidisciplinary design process of structures by allowing the designer to carry out various analysis and design tasks in parallel. The tasks can belong to an individual discipline as well as to other disciplines (such as in multidisciplinary optimization problems).

Special-Purpose Computing Hardware

With the continued reduction in the cost of computer hardware, a number of special-purpose computers have emerged that offer increased speed for specific problems when compared with general-purpose computers. Examples are provided by the Navier-Stokes Computer (NSC) being developed at Princeton University. The NSC is a parallel processing machine comprised of a multi-dimensional array of processing nodes, each with a local memory. Its primary function is the direct numerical simulation of complex flows. A description of the NSC machine is given in [29].

There is a growing trend of moving from software-based processing to hardware-based processing, and it is likely that the classical linear finite-element analysis will become a hardware function in the future.

Small Systems

A broad spectrum of low-cost small systems exist now, including new powerful microprocessors, transputers, embedded computers, portable computers, laptop computers, desktop computers, engineering workstations, minis and superminis. Several classifications have been attempted for these systems based on word length

(8-bit, 16-bit and 32-bit machines), cost, amount of directly addressable memory, and computing speed (4, 6, 16, 20, and 25 MHz - see, for example, [30]). However, the dramatic increase in hardware capabilities of small systems coupled with the rapid reduction in cost makes these classifications of questionable value. Herein, the development in microprocessors, transputers, and engineering workstations are discussed.

Microprocessors and Chip Technology. The trend of ever-increasing the number of devices packaged on a chip has resulted in the miniaturization and increase in speed of microprocessors and minicomputers. The new powerful chips can be used as monitoring systems (e.g., embedded computers) for the detection, recording and evaluation of stochastic damage, thereby increasing the mean time between inspection for structural components.

RISC Processors and Transputers. In the 1970's the efforts directed towards creating machines with very fast clock cycle, that can execute instructions at the rate of one per cycle (like microprogrammed controllers), resulted in the development of Reduced Instruction-Set Computing (RISC) processors. These are microprocessors that provide high-speed execution of simple instructions. The implementation of RISC architecture began at IBM in the mid 1970's. RISC architecture was used in the development of high-performance superminis (e.g., PYRAMID 90X and RIDGE 32 computers). The basic notion of RISC has now evolved to encompass chips in which the chip areas formerly used for decoding, and executing complex instructions, are used for caching instructions. RISC chips offer the following advantages over the Complex Instruction-Set Computing (CISC) chips: much smaller size chips, more throughput, shorter design time, better support for high-level languages, and the ability to emulate other instruction sets.

In the last few years a new type of powerful VLSI chip (superchip) which packs RISC and a limited amount of RAM has been developed. The superchip is called a *transputer*, and is manufactured by Inmos, Ltd. of Bristol, England. The transputer was designed from the outset for parallel and distributed processing. The transputer has hardware support for parallel tasks, including local (on-chip) memory, and four high-speed communication links. These can transfer data at the rate of 10-20 Mb (Megabits) per second, thereby enabling the transputer to be interconnected in powerful arrays. A high-level programming language, Occam, has been especially developed for parallel processing on the transputer. Also, other languages (e.g., Fortran, Pascal and C) are available for use on the transputer.

A range of transputer boards and modules now exist which can be plugged into conventional and desktop computers to speed up the computations. Also, a desktop supercomputer with many transputers has been developed by Meiko, a company in Bristol, England, that grew out of Inmos. Transputer networks, formed by linking together transputers in arrays, pipelines, rings and other patterns, have been efficiently used in a wide variety of applications including solution of finite element equations and graphics processing. To date the latest transputer, the T800-20 is a 32-bit transputer with four serial links, 4K of on-chip memory, and a built-in floating point unit. It has an execution rate of 10 MIPS and a computational speed of 1.5 MFLOPS. It is equivalent to a VAX 8600 on a single chip.

Engineering Workstations with Advanced Visualization Capabilities. Single-user engineering workstations using VLSI 32-bit processor chips, and having internal graphics-processing facilities, and 50 MBytes (or more) of addressable memory have been developed. Examples of such systems are the HP-9000, series 800 and Turbo/SRX, and the Apollo computers (10000 series). A superstation with a peak performance of 40 MFLOPS has been developed by FPS Computing (Model 300). It is anticipated that in the next few years, the powerful workstations (designated 5M machines) will be desktop computers with sustained speed of 100 MFLOPS, over 1 MIP execution rate, over 100 MBytes of addressable memory, over megapixel display, and over megabit transmission rate within local area networks (LAN's). Future advanced visualization facilities include high-resolution and high-speed graphics, video and film animation capabilities.

ADVANCES IN PROGRAMMING ENVIRONMENT AND SOFTWARE TECHNOLOGY

Although considerable effort is now devoted to increasing the productivity of the analyst and designer through the development of powerful programming environment, software and programming languages remain the primary pacing items for exploiting the potential of new computing systems.

Programming Environment

This refers to the array of physical and logical means by which the analyst transmits instructions to the machine. It includes the user-interface devices; interactive programming tools (e.g., debuggers, editors, file-maintenance utilities), and tools for automatic (and semiautomatic) mapping of numerical algorithms on different machines.

The programming environments on most of the existing new computing systems are limited to a standard sequential language compiler, and extensions to support concurrency (viz. vectorization and parallelization). The extensions of currently used programming languages to support concurrency (Fortran, C, Lisp) are not the same on different machines. A description of recent work in development of software environment is given in [31, 32, 33 and 34].

Current and future interface devices have been discussed in a previous section. Future powerful programming languages should enable the user to state the mathematical and logical formulation of the problem in the expectation that the language can fill in the details. These languages should be architecture-independent high-level languages to allow the portability of the programs.

AI Knowledge-Based and Expert Systems

AI-based expert systems, incorporating the experience and expertise of practitioners, have high potential for the modeling, analysis and design of structures. These systems can aid the structural analyst in the initial selection and adaptive refinement of the model, as well as in the selection of the appropriate algorithm used in the analysis. Expert systems can also aid the structural designer by freeing him from such routine tasks as the development of process and material specifications. A review of the capabilities of some of the currently available expert systems and their limitations are given in [35 and 36]. A description of a knowledge-based system used as a modeling aid for aircraft structures is given in [37].

Large Powerful Data Management Systems and Databases

Future engineering software systems are likely to have the basic analysis software (such as data management, control, etc.) as part of the software infrastructure and the discipline specifics (such as the finite element properties of the structure) as part of application software. Conventional database management systems (DBMs) developed in the last decade such as relational database management system (RIM), provided multidisciplinary coordination, and helped in the integration of structural analysis programs into CAD/CAM systems. However, these conventional DBMs do not meet the data requirements for the current and emerging engineering/design environment (see [38]). Among the different advanced data/process modeling methodologies which have high potential for multidisciplinary analysis and design applications are: the three-level IDEF methodology developed by the Air Force's integrated computer-aided manufacturing program (see [39]); Nijssen's information analysis method based on a binary relationship model; Entity relationship model; and object-oriented data model. A description of these models is given in [38].

ADVANCES IN NUMERICAL ALGORITHMS AND COMPUTATIONAL STRATEGIES

To achieve high performance from any computing system, it is necessary either to tailor the computational strategy and numerical algorithms to suit the architecture of the computer, or to select the architecture which may effectively map the computational strategy and execute the numerical algorithm. Extensive work has been devoted to the development of vectorized and parallel numerical algorithms for new computing systems. Review of some of this work is contained in a number of monographs and survey papers.

In this section a brief review is given of the recent progress made in special parallel numerical algorithms and computational strategies that are influencing structures calculations.

Parallel Numerical Algorithms

In parallel algorithms, independent computations are performed in parallel (i.e., executed concurrently). To achieve this parallelism, the algorithm is divided into a collection of independent tasks (or task modules) which can be executed in parallel and which communicate with each other during the execution of the algorithms. Parallel algorithms can be characterized by the following three factors:

1. Maximum amount of computation performed by a typical task module

before communication with other modules.

2. Intermodule communication topology, which is the geometric layout of the network of task modules

3. Executive control to schedule, enforce the interactions among the different task modules and ensure the correctness of the parallel algorithm.

The three aforementioned factors have been used in [40] as a basis for classifying parallel algorithms on the conceptual level, and for relating each parallel algorithm to the parallel (or pipeline) architecture to which it naturally corresponds.

The design of a parallel algorithm must deal with a host of complex problems, including data manipulation, storage allocation, memory interference, and in the case of parallel processors, interprocessor communication. In general, the parallel numerical algorithms reported in the literature fall into two categories: reformulation (or restructuring) of serial algorithms into concurrent algorithms, and algorithms developed especially for parallel machines.

Most of the work on parallel numerical algorithms belongs to the first category, i.e., decomposition of familiar serial algorithms into concurrent tasks. Examples are matrix operations, direct and iterative methods for solution of algebraic equations, and eigenvalue extraction techniques (see, for example, [41, 42, 43, 44 and 45]).

The second category includes the algorithms whose development was spurred by performance criteria for parallel processing. These algorithms have been referred to as *uniquely parallel* and only a few of them have been reported in the literature [46]. Examples of uniquely parallel algorithms are provided by the parallel superconvergent multigrid method [47] and the fully parallel algorithm for symmetric eigenvalue problem [48]. In some cases the performance of uniquely parallel algorithms is superior to their serial counterparts.

Construction of Parallel Algorithms

The development of parallel numerical algorithms generally follows one or both of two related approaches: *reordering*, and *divide and conquer*. Reordering refers to restructuring the computational domain and/or the sequence of operations in order to allow concurrent computations. For example, the order in which the nodes of a finite element grid are processed may change the degree of parallelism that can be achieved in the solution of the resulting algebraic equations ([49]).

The divide and conquer approach involves breaking a task up into smaller subtasks that can be treated independently. The degree of independence of these tasks is a measure of the effectiveness of the numerical algorithm since it determines the amount and frequency of communication and synchronization. This idea pervades many of the parallel algorithms and can be extended to the overall computational strategy as described in the succeeding section.

Comments on Parallel Algorithms and Their Implementation

The following comments concerning parallel algorithms and their implementation are in order:

1. Effective parallel algorithms are not necessarily effective on sequential computers. In fact, some parallel algorithms involve additional (redundant) floating point operations which make them inefficient on sequential machines. Also, restructured serial algorithms may not be the most efficient on parallel processing machines.

2. The mathematical properties of the serial and parallel implementations of the same algorithm can be different. For example, the rate of convergence and numerical stability of serial and parallel iterative techniques can be different. In some cases parallel implementation can degrade the performance and in other cases, it improves it ([46]).

3. To achieve high performance both the numerical algorithm and its implementation must be carefully tailored to the particular machine being used. This raises the question of portability of parallel programs. It is not practical to develop algorithms and programs for each new computer. Also, it is not desirable to achieve portability at the expense of performance. A number of studies have been devoted to achieving high performance and portability of numerical algorithms on advanced computers. Two approaches have been proposed in [50] and [51]: a) restructuring of algorithms in terms of high-level modules (e.g., matrix-matrix and matrix-vector operations); and b) developing and implementing an abstraction of parallel processing that is independent of the architecture.

4. On most of the currently-available multiprocessing systems vectorization offers greater performance improvement over multitasking (i.e., parallelization). Consequently, if multitasking conflicts with efficient vectorization (e.g., multitasking results in short vector lengths), then the algorithm should be vectorized rather than parallelized.

Performance of Parallel Numerical Algorithms

Computational complexity (e.g., number of floating-point arithmetic operations) has long been used as a measure of the performance of serial algorithms. However, it is not an appropriate measure for parallel numerical algorithms. This is because parallel computers can support extra computation at no extra cost if the computation can be organized properly; and parallel computers are subject to new overhead costs (e.g., synchronization and communication) that are not reflected by computational complexity.

One of the most commonly-used measures for the performance of parallel numerical algorithms is the speedup, S, which is defined as follows:

$$S = \frac{\text{execution time using one processor}}{\text{execution time using } p \text{ processors}}$$

The measure S has the advantage that it uses the execution time and, therefore, incorporates the synchronization and communication overhead. However, it has the drawback of comparing the execution time of the same algorithm on the single and multiple processors.

Another definition of speedup, based on Amdahl's Law, was proposed by Ware [52], and is expressed by the following simple formula:

$$S(p,f) = \frac{1}{1-f\left(1-\frac{1}{p}\right)} \qquad (1)$$

where S is the maximum speedup achievable by using p processors; and f is the fraction of computational work done in parallel (at the high execution rate).

In Eq. 1 the execution time using a single processor has been normalized to unity. The rate of change of S with f, at f=1, is quadratic in p, i.e.,

$$\frac{dS}{df}\Big|_{f=1} = p^2 - p \qquad (2)$$

Therefore, for massively parallel processors the fraction of parallelism must grow with the number of processors in order to achieve reasonable speedups ([45]).

The utilization rate of the multiprocessor system, U, is defined as follows:

$$U(p,f) = \frac{S}{p} \qquad (3)$$

A utilization rate of 1 means that every processor is busy computing all the time.

Figure 10 shows the theoretical speedup and the utilization rate of multiprocessor systems as a function of the fraction of parallelism, f, and the number of processors, p. The figure illustrates a key issue in multiprocessor machines: as the number of processors increases, then for a given fraction of parallelism, the degree of utilization decreases.

The following comments can be made regarding Amdahl's Law and Ware's model:

1. Ware's model assumes that the parallel processing machine is a two-state machine in the sense that at any instant of time either all the processors are operating or only one of them is operating.

2. Amdahl's Law can be extended to computers with more than two modes of operation with one mode having a lower rate of execution than others. For example, if the scalar mode is taken as the low rate and a balanced higher rate, representative of vector, memory and I/O is taken as the high rate, then Eq. 1 can be used to give the maximum speedup achievable by the system ([23]).

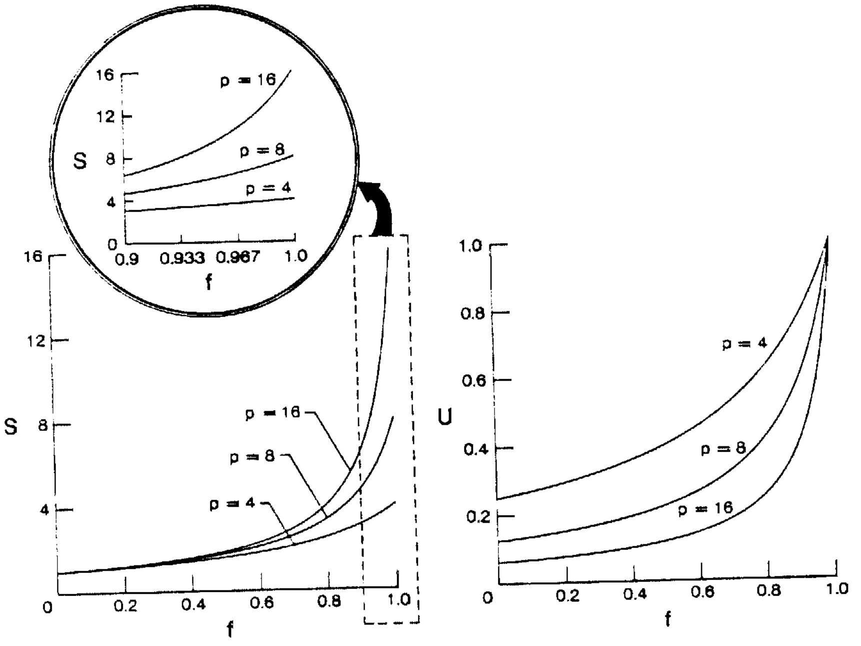

Figure 10 - Theoretical speedup, S, and utilization rate, U, of multiprocessor systems as a function of the fraction of parallelism, f, and the number of processors, p.

3. Ware's model does not account for the overhead associated with inter-processor communication, synchronization (for controlling data access and for program control), among others. This overhead may increase with increasing the number of processors, resulting in a speed-down behavior ([46 and 53]).

4. Eq. 1 measures the speedup relative to the implementation on a single processor of the same algorithm. It does not necessarily measure the efficiency gain due to parallelization. This will be discussed further in the succeeding sections.

5. In Ware's model the implicit assumption is made that f is independent of p, which is only true if the problem size is fixed. However, in practice the size of the problem increases with the increase in the number of available processors. The parallel part of the program scales with the problem size, but the times for the

program loading, I/O, and serial computations do not usually scale with problem size. A discussion of the effect of problem size on the performance of parallel algorithms is discussed in [54] and [55].

6. Ware's model can be used for estimating the speedup due to vectorization on vector machines, if f is interpreted as the fraction of vectorizable work, i.e., the maximum speedup due to vectorization is given by:

$$S_v = \frac{1}{1 - f_v\left(1 - \frac{1}{R_v}\right)} \tag{4}$$

where f_v is the fraction of vectorizable work, and R_v is the ratio of the vector to the scalar execution rate. Note that Eq. 4 does not account for the effect of overlapping scalar and vector operations (which can be done on some vector processors).

7. The maximum speedup due to the combined effects of parallel execution and vectorization (i.e., parallel vectorization) can be represented by the product of $S(p,f)$ (Eq. 1), and S_v (Eq. 4). The speedup is depicted in Fig. 11 as a function of the fractions of vectorizable and concurrent work, f_v and f_p, for a vector multiprocessor machine with four processors and an R_v of ten (vector execution rate ten times that of the scalar execution rate). Note that when $f_v = f_p = 0.9$ the speedup is only 16.19 (less than 41% of the theoretical maximum speedup).

Special Computational Strategies
Table 5 lists the different phases involved in the finite element static analysis, and
their suitability for vectorization and parallelization. A number of special strategies
can be used to increase the degree of parallelism and/or vectorization in finite element
computations. These strategies are applications of the principle of *divide and con-*
quer, based on breaking a large (and/or complex) problem into a number of smaller
(and/or simpler) subproblems which may be solved independently on distinct proces-
sors. The degree of independence of the subproblems is a measure of the effective-
ness of the algorithm since it determines the amount and frequency of communication
and synchronization.

Herein, three strategies are discussed: domain decomposition and substructuring; operator splitting; and element-by-element strategies.

1. *Domain decomposition and substructuring*. The basic idea of domain (or
spatial) decomposition is to divide the domain into a number of (possibly overlap-
ping) regions. The initial/boundary-value problem is decomposed into one that
involves solution of initial/boundary-value problems on subdomains, thereby intro-
ducing spatial parallelism. Since the solution is not available at the interfaces
between regions, it is modified iteratively as part of the solution procedure. A review
of parallel domain decomposition techniques is given in [56].

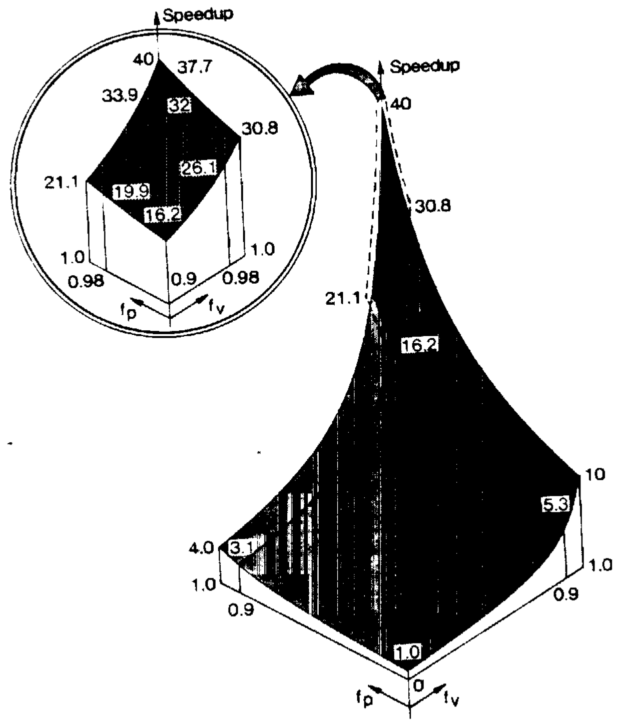

Figure 11 - Theoretical speedup on a vector multiprocessor as a function of the fractions of parallelism and vectorization f_p and f_v

Substructuring techniques are closely related to domain decomposition. They can also be identified at the algebraic level by partitioning the associated matrices in an appropriate way to separate the degrees of freedom that are to be eliminated (the internal degrees of freedom in different substructures) from those to be retained (interface degrees of freedom). Substructuring techniques lend themselves directly to parallel vectorization ([49] and [57]). However, the partitioning of a discretized structure into substructures to achieve well-balanced workload distribution among the different processors is a difficult combinatorial problem. A simulated annealing algorithm for the approximate solution of this problem is presented in [58]. The algorithm is analogous to a method used in statistical mechanics for simulating the annealing process in solids. Other partitioning strategies are described in [59] and [60].

2. *Operator splitting*. The notion of splitting has long been used to synthesize the solution of a complicated problem from that of a simpler problem (or a sequence of simpler problems). Among the different applications of splitting are the breaking

of a multidimensional problem into a sequence of one-dimensional problems, and the development of iterative (and semi-iterative) techniques for solution of algebraic equations. Splitting can be used as a means of partitioning the computational task into a number of subtasks that are either independent, or only loosely coupled, so that the computations can be made on distinct processors with little communication and sharing.

Table 5 - Different phases of finite element structural analysis steady-state (static) problems

Phase	Suitability for Parallelization/ Vectorization
• Input problem characteristics, element and nodal data, and geometry	• Can be parallelized
• Evaluation of element characteristics	• Easy to parallelize and can be vectorized
• Assembly	• Require special care for parallelization • Difficult to vectorize
• Incorporation of boundary conditions	• Easy to parallelize
• Solution of algebraic equations	• Important to vectorize and parallelize
• Postprocessing	• Can be parallelized and vectorized

3. *Element-by-element solution strategies.* The modular element-by-element logic inherent in the finite element analysis procedure has been used to develop solution strategies which do not require the explicit generation of the global stiffness matrix. The frontal elimination method was originally proposed by Irons [61] to bypass the assembly process. In recent years element-by-element strategies have been developed for the solution of static and dynamic problems as well as adaptive grid generation. A review of these strategies is given in [62]. Element-by-element strategies are well-suited to vector and parallel processing and, therefore, should be

seriously considered for use in parallel processing machines.

EFFECTIVE STRATEGY FOR SOLUTION OF LARGE-SCALE STRUCTURAL PROBLEMS ON NEW COMPUTING SYSTEMS

A strategy is being developed for the efficient analysis of large complex structures on new computing systems. The strategy is designed for multiprocessor computers with either: 1) a shared memory and a small number of powerful processors such as the CRAY X-MP, CRAY-2 or CRAY-3; or 2) a hierarchical memory and a small number of clusters of medium-range processors such as the Cedar Project of the University of Illinois at Urbana-Champaign (with up to four clusters of Alliant FX/8 processors).

The strategy aims at maximizing the degree of parallelism at different levels of the finite element analysis process, including: 1) formulation level (through the use of mixed finite element models); 2) analysis level through the application of a novel spatial partitioning (or substructuring) technique based on symmetry transformations; 3) numerical algorithm level through the combined use of operator splitting techniques, iterative processes, and carefully tailored numerical algorithms for the particular hardware; and 4) implementation level through effective combination of vectorization, multitasking and microtasking, whenever available. The strategy is outlined subsequently.

Basic Idea and Key Elements of the Strategy

The foregoing strategy is based on approximating the response of the structure by a combination of symmetric and antisymmetric response vectors (or modes). The strategy can also be thought of as generating the response of the original unsymmetric structure using *large perturbations* from that of a simpler structure in which the symmetric and antisymmetric components of the response are uncoupled. The symmetric and antisymmetric modes are, therefore, generated using a reduced-size model of the modified structure (1/m the size of the original finite element model, where m is an integer which depends on the number of symmetry operations used).

The three key elements of the strategy are: 1) use of mixed (primitive variable) formulation with independent shape functions for different fields, and with all but the displacement field allowed to be discontinuous at interelement boundaries; 2) operator splitting, or restructuring of the governing discrete equations of the structure to delineate the symmetric and antisymmetric vectors constituting the response; and 3) use of preconditioned conjugate gradient (PCG) technique to generate the symmetric and antisymmetric response vectors.

Comments on the Foregoing Strategy

The following two comments regarding the strategy are in order:

1. The application of the foregoing strategy to structural problems can be divided into two phases, each of which is well-suited for parallel processing: a) preprocessing phase in which the finite element model is partitioned into substructures and reduced arrays are generated (whose dimension is 1/m times the total number of degrees of freedom of the original finite element model); b) solution phase

in which the PCG technique is used to generate the response of the original structure. It is desirable to select the symmetries such that m equals the number of processors, p. The physical interpretation of the preprocessing phase is depicted in Figs. 12 and 13 for the case m=4. The application of the foregoing strategy to linear stress analysis, free vibration, and nonlinear dynamic problems is described in [63], [64] and [65], respectively, for the case m=2. The case m=4 is described in [66].

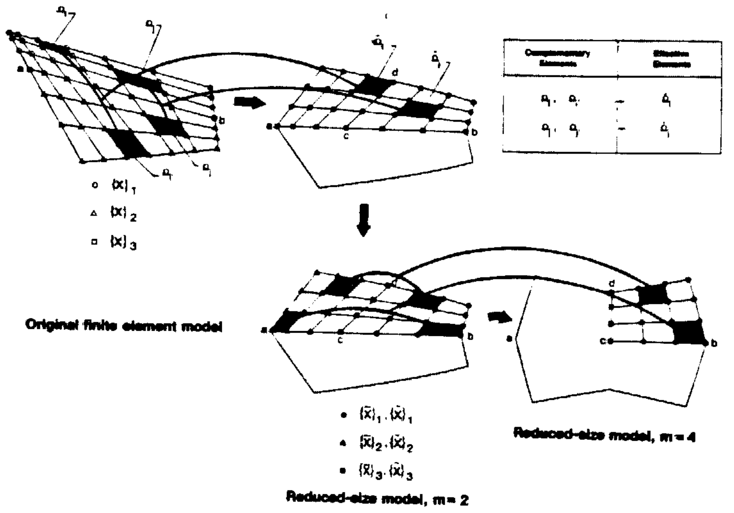
Original finite element model
Reduced-size model, m = 2
Reduced-size model, m = 4
Complementary Elements
Effective Elements

Figure 12 - Original finite element model and reduced-size models

2. The decomposition of the response vector into symmetric and antisymmetric components can be performed by means of matrix transformations. For the case m=2, the matrix transformations are given in [63] and [65].

Performance Evaluation of the Foregoing Strategy

The strategy has been implemented on the CRAY X-MP/416 at CRAY Research, Inc. in Mendota Heights, Minnesota, and the Alliant FX/8 computers at the University of Illinois at Urbana-Champaign. Herein, the efficiency gain resulting from the use of the strategy is discussed for a typical nonlinear dynamic problem of a laminated anisotropic panel with an off-center circular cutout. The loading is assumed to be uniformly distributed and normal to the panel surface and to have a step variation in time. The panel is made of graphite-epoxy material (see Fig. 14). Mixed finite element models were used for the spatial discretization, and implicit three-step method was used for the temporal integration. The details are given in [66]. Normalized contour plots for the displacement and velocity components at 3.0 msec., drawn on the undeformed middle surface of the panel, are shown in Fig. 15.* The measured CP times and processing rates obtained using the present strategy on a single CPU are shown in Fig. 16. Comparison of the wall-clock times obtained by the present strategy on one-, two- and four- CPUs, with those of the direct analysis of the panel

*See Figure 15 in colour section, page 159.

(with no partitioning) are given in Table 6. As can be seen from Table 6, the use of the present strategy on four CPU-machine reduces the total analysis time by nearly an order of magnitude, compared with that required by the direct analysis (on a single processor).

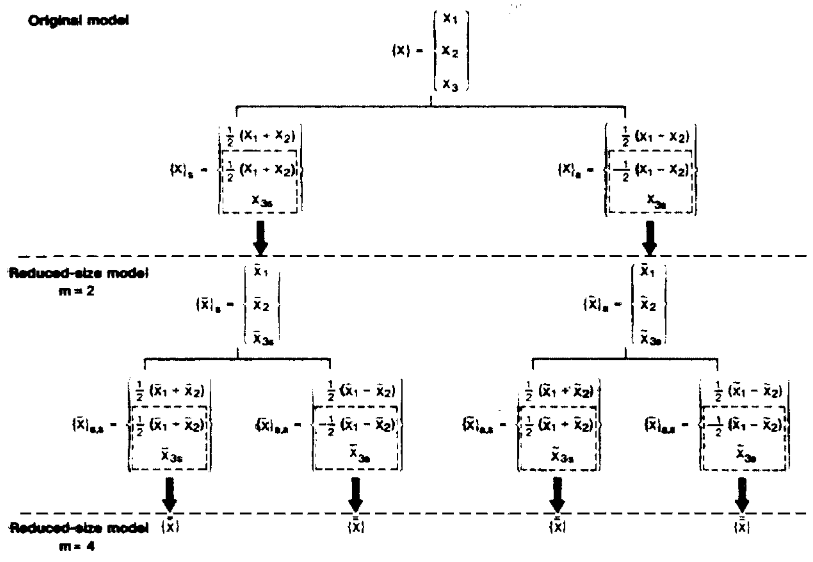

Figure 13 - Fundamental unknowns in the original and reduced-size models (see Fig. 12)

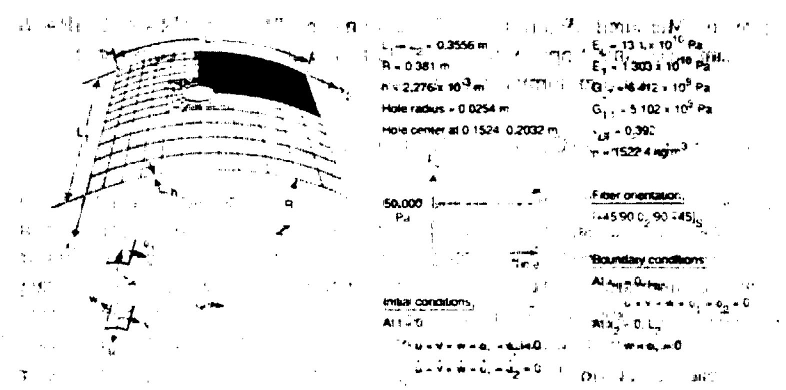

Figure 14 - Laminated anisotropic composite panel with off-center circular cutout used in the present study

Module no.	Description
1.	Preprocessing
2.	Generation of nonlinear elemental contributions and assembling R.H.S.
3.	Eliminating stresses from L.H.S. and assembly of L.H.S.
4.	Incorporation of B.Cs. and decomposition
5.	Eliminating stresses from R.H.S.
6.	Back solve
7.	Recovery of stresses
8.	Step lengths and conjugate search directions
9.	Convergence checks

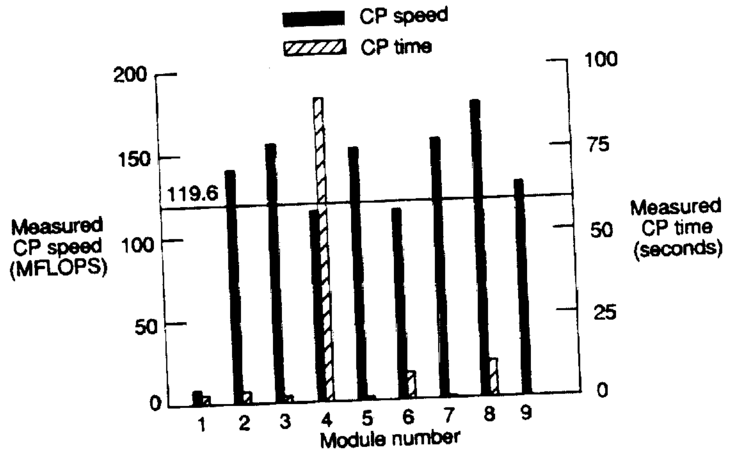

Figure 16 - Measured CP times and CP speed on one CPU of the CRAY X-MP/416. Laminated anisotropic panel with off-center circular cutout subjected to uniform normal loading p_o=-50,000 Pa (see Fig. 14).

ORIGINAL PAGE IS OF POOR QUALITY

SUMMARY AND CONCLUDING REMARKS

A review is given of the recent advances in computer technology that are likely to impact structural analysis and design. The computational needs for future structures technology are described. The characteristics of new and projected computing systems are summarized. At one end of the spectrum there are the top-of-the-range large supersystems such as CRAY-2, ETA-10 and CRAY Y-MP. The performance of large supersystems will continue to improve and their peak computational speed is likely to reach a teraflop (1×10^{12} floating point operations per second) before the end of the present century. These supersystems will make possible new levels of sophistication in structural modeling as well as in problem depth and scope which were not possible before. At the other end of the spectrum, embedded computers will aid in the control of the devices they reside in and will help in the realization of intelligent (smart or adaptive) structures. The emerging small low-cost engineering workstations

and personal computers will provide a high degree of interactivity and free the analysts from the constraints that are often imposed on them by large centralized computation centers. Intelligent interfaces allowing multiple media interaction for both input and output (e.g., graphics and natural languages) will facilitate the user-machine communication. Flexible high-capacity networks will allow collaborative computing by linking structural analysts and designers at different locations.

Advances in programming environments, numerical algorithms, and computational strategies for new computing systems are reviewed, and a novel partitioning strategy is outlined for maximizing the degree of parallelism. The strategy is designed for computers with a shared memory and a small number of powerful processors (or a small number of clusters of medium-range processors). It is based on approximating the response of the structure by a combination of symmetric and antisymmetric response vectors, each obtained using a fraction of the degrees of freedom of the original finite element model. The strategy was implemented on the CRAY X-MP/4 and the Alliant FX/8 computers. For nonlinear dynamic problems on the CRAY X-MP with four CPUs, it resulted in an order of magnitude reduction in total analysis time, compared with the direct analysis on a single-CPU CRAY X-MP machine.

Table 6 - Performance evaluation of presented strategy on the CRAY X-MP/416 at CRAY Research, Inc. in Mendota Heights, Minnesota

	Full Structure (*Optimized Code*) (160 MFLOPS)	Partitioned Structure (*Nearly Optimized Code*) (119.6 MFLOPS)
Number of degrees of freedom	3818 displacements 6144 stresses	971 displacements 1536 stresses
Semibandwidth of equations	700	315
Wall clock time for first ten steps, (sec.)	319.7	122.6 one processor 66.7 two processors 36.3 four processors
Speedup	1.0	2.61 one processor 4.79 two processors 8.81 four processors

The discussion of the new computing systems presented herein is intended to give structural analysts and designers some insight into the potential of these systems for providing cost-effective solutions of complex structural problems, and to stimulate research and development of the necessary numerical algorithms, firmware and software to realize this potential.

ACKNOWLEDGMENT

The present work is supported by an Air Force Office of Scientific Research Grant No. AFOSR-88-0136, and a NASA Grant No. NAG1-730.

REFERENCES

1. Torrero, E. A. (Ed.). Next-Generation Computers, Spectrum Series, IEEE, New York, 1985.
2. Cole, B. C. Here Comes the Billion-Transistor IC, Electronics, Vol. 60, No. 7, pp. 81-85, 1987.
3. Meindl, J. D. Chips for Advanced Computing, Scientific American, Vol. 257, No. 4, pp. 78-89, 1987.
4. Sluss, J. J., Veasey, D. L., Batchman, T. E. and Parrish, E. A. An Introduction to Integrated Optics for Computing, Computer, IEEE, Vol. 20, No. 12, pp. 9-23, 1987.
5. Tank, D. W. and Hopfield, J. J. Collective Computation in Neuronlike Circuits, Scientific American, Vol. 257, No. 6, pp. 104-114, 1987.
6. Noor, A. K., Storaasli, O. O. and Fulton, R. E. Impact of New Computing Systems on Computational Mechanics and Flight-Vehicle Structures Technology, in AGARD Report No. 706, The Influence of Large-Scale Computing on Aircraft Structural Design, AGARD Structures and Materials Panel Meeting, Sienna, Italy, April 2-4, 1984.
7. Noor, A. K., Storaasli, O. O. and Fulton, R. E. Impact of New Computing Systems on Finite Element Computations. Part 4, Chapter 4, Finite Element Handbook, (Ed. by H. Kardestuncer and D. H. Norrie), pp. 4.230-4.233, McGraw-Hill, New York, 1987.
8. Noor, A. K. and Atluri, S. N., Advances and Trends in Computational Structural Mechanics, AIAA Journal, Vol. 25, No. 7, July 1987, pp. 977-995.
9. Wollard, K., Solid State, IEEE Spectrum, Vol. 25, No. 1, pp. 44-45, 1988.
10. Hwang, K. and Briggs, F. A. Computer Architecture and Parallel Processing, McGraw-Hill, New York, 1984.
11. Kryder, M. H. Data-Storage Technologies for Advanced Computing, Scientific American, Vol. 257, No. 4, pp. 117-125, 1987.
12. Foley, J. D. Interfaces for Advanced Computing, Scientific American, Vol. 257, No. 4, pp. 127-135, 1987.
13. Kahn, R. E. Networks for Advanced Computing, Scientific American, Vol. 257, No. 4, pp. 136-143, 1987.
14. Flynn, M. J. Some Computer Organizations and Their Effectiveness, IEEE Transactions on Computers, Vol. C-21, pp. 948-960, 1972.
15. Hwang, K. Advanced Parallel Processing with Supercomputer Architectures, Proceedings of the IEEE, Vol. 75, No. 10, pp. 1348-1379, 1987.

16. Hack, J. J. Peak vs. Sustained Performance in Highly Concurrent Vector Machines, Computer, IEEE, Vol. 19, No. 9, pp. 11-19, 1986.
17. Dongarra, J. J., Martin, J. and Worlton, J. Computer Benchmarking: Paths and Pitfalls, IEEE Spectrum, Vol. 24, No. 7, pp. 38-43, 1987.
18. Amdahl, G. The Validity of the Single-Processor Approach to Achieving Large-Scale Computing Capabilities, Vol. 30, pp. 483-485, Proceedings of the American Federation of Information Processing Societies, Washington, D.C., 1967.
19. Amdahl, G. M. Limits of Expectation, International Journal of Supercomputer Applications, Vol. 2, No. 1, 1988, pp. 88-94.
20. Dongarra, J. J. and Duff, I. S. Advanced Architecture Computers. Technical Memorandum No. 57 (Rev. 1), Argonne National Laboratory, Argonne, Illinois, 1987.
21. McBryan, O. A. State of the Art in Highly Parallel Computer Systems, in Parallel Computations and Their Impact on Mechanics (Ed. A. K. Noor), pp. 31-47, Proceedings of the Symposium on Parallel Computations and Their Impact on Mechanics, Boston, MA, Dec. 1987, AMD Vol. 86, American Society of Mechanical Engineers, New York, 1987.
22. Noor, A. K. Parallel Processing in Finite Element Structural Analysis, Engineering with Computers, Vol. 3, pp. 225-241, 1988.
23. Worlton, J. Toward a Science of Parallel Computation, in Computational Mechanics - Advances and Trends (Ed. A. K. Noor), pp. 23-35, Proceedings of the Symposium on Future Directions of Computational Mechanics, Anaheim, CA, Dec. 1986, AMD Vol. 75, American Society of Mechanical Engineers, New York, 1986.
24. Fortes, J. A. B. and Wah, B. W. Systolic Arrays - From Concept to Implementation, Computer, IEEE, Vol. 20, No. 7, pp. 12-17, 1987.
25. Hayes, L. J. Systolic Arrays for Finite Element Calculations, in Parallel Computations and Their Impact on Mechanics (Ed. A. K. Noor), pp. 229-238, in Proceedings of the Symposium on Parallel Computations and Their Impact on Mechanics, Boston, MA, Dec. 1987, AMD Vol. 86, American Society of Mechanical Engineers, New York, 1987.
26. Law, K. H. Systolic Arrays for Finite Element Analysis, Computers and Structures, Vol. 20, Nos. 1-3, pp. 55-65, 1985.
27. Melhem, R. G. On the Design of a Pipelined/Systolic Finite Element System, Computers and Structures, Vol. 20, Nos. 1-3, pp. 67-75, 1985.
28. Moore, W., McCabe, A. and Urquhart, R. (Eds.). Systolic Arrays - Papers Presented at the First International Workshop on Systolic Arrays, Oxford, July 2-4, 1986, Adam Hilger, Bristol and Boston, 1986.
29. Hayder, M. E., Flannery, W. S., Littman, M. G., Nosenchuck, D. M. and Orszag, S. A., Large-Scale Turbulence Simulation on the Navier-Stokes Computer, Computers and Structures, Vol. 30, No. 1/2, pp. 357-364, 1988.
30. Conaway, J. H. Structural Engineering Software on Small Computers, American Society of Mechanical Engineers, New York, Paper 80-C2/Aero-7, 1980.
31. Jordan, H. Structuring Parallel Algorithms in an MIMD, Shared Memory Environment, Parallel Computing, Vol. 3, pp. 93-110, 1986.

32. Nicolau, A. A Development Environment for Scientific Parallel Programs, Applied Mathematics and Computation, Vol. 20, Nos. 1-2, pp. 175-183, 1986.
33. Pratt, T. W. The PISCES 2 Parallel Programming Environment, NASA CR-178327, 1987.
34. Gannon, D., Atapattu, D., Lee, M. H. and Shei, B. A Software Tool for Building Supercomputer Applications, in Parallel Computations and Their Impact on Mechanics (Ed. A. K. Noor), pp. 81-92, in Proceedings of the Symposium on Parallel Computations and Their Impact on Mechanics, Boston, MA, Dec. 1987, AMD Vol. 86, American Society of Mechanical Engineers, New York, 1987.
35. Dym, C. L. Expert Systems: New Approaches to Computer-Aided Engineering, Part I, pp. 99-115, in Proceedings of the AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics and Materials Conference, Palm Springs, CA, May 1984.
36. Mackerle, J. and Orsborn, K. Expert Systems for Finite Element Analysis and Design Optimization - A Review, Engineering Computations, Vol. 5, No. 2, pp. 90-102, 1988.
37. Taig, I. C. Expert Aids to Reliable Use of Finite Element Analysis, in Reliability of Methods for Engineering Analysis (Ed. K. J. Bathe and D. R. J. Owen), pp. 457-474, Pineridge Press, Swansea, United Kingdom, 1986.
38. Fulton, R. E. and Yeh, C.-p. Managing Engineering Design Information, in Proceedings of the AIAA//AHS/ASEE Aircraft Design, Systems and Operations Conference, Atlanta, GA, Sept. 7-9, 1988.
39. Parks, C. H. Tutorial: Reading and Reviewing the Common Schema for Electrical Design and Analysis, in Proceedings of 24th ACM/IEEE Design Automation Conference, pp. 479-483, 1987.
40. Kung, H. T. The Structure of Parallel Algorithms, Advances in Computers, Vol. 19, pp. 65-112, Academic Press, 1980.
41. Schendel, U. Introduction to Numerical Methods for Parallel Computers, Ellis Horwood, Ltd., United Kingdom, 1984.
42. Dongarra, J. J. and Sorensen, D. C. Linear Algebra on High-Performance Computers, in Parallel Computing 85 (Eds. M. Feilmeier, G. Joubert and U. Schendel), pp. 3-32, Elsevier, New York, 1985.
43. Adams, L. Reordering Computations for Parallel Execution, Communications in Applied Numerical Methods, Vol. 2, pp. 263-271, 1986.
44. McBryan, O. and Van de Velde, E. Matrix and Vector Operations on Hypercube Parallel Processors, Parallel Computing, Vol. 5, pp. 117-125, 1987.
45. Ortega, J. M. Introduction to Parallel and Vector Solution of Linear Systems, Plenum Press, New York, 1988.
46. Buzbee, B. L. Uniquely Parallel Algorithms, in Parallel Computations and Their Impact on Mechanics (Ed. A. K. Noor), pp. 95-100, in Proceedings of the Symposium on Parallel Computations and Their Impact on Mechanics, Boston, MA, Dec. 1987, AMD Vol. 86, American Society of Mechanical Engineers, New York, 1987.
47. Frederickson, P. and McBryan, O. Parallel Superconvergent Multigrid, in Multigrid Methods - Theory, Applications and Supercomputing, Lecture Notes in Pure and Applied Mathematics, Vol. 110, (Ed. S. McCormick), Marcel Dekker, New York, pp. 195-210, 1988.

48. Dongarra, J. J. and Sorensen, D. C. A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem, SIAM Journal of Scientific and Statistical Computing, Vol. 8, No. 2, pp. 139-154, 1987.
49. Adams, L. and Voigt, R. A Methodology for Exploiting Parallelism in the Finite Element Process, in Proceedings of the NATO Workshop on High Speed Computations, NATO ASI Series (Ed. J. Kowalik), Springer, Berlin, F-7, pp. 373-392, 1984.
50. Dongarra, J. J. and Sorensen, D. C. SCHEDULE: Tools for Developing and Analyzing Fortran Programs, MCS-TM-86, Argonne National Laboratory, Argonne, Illinois, Nov. 1986.
51. Boyle, J., Butler, R., Disz, T., Glickfeld, B., Lusk, E., Overbeek, R., Patterson, J. and Stevens, R. Portable Programs for Parallel Processors, Holt, Rinehart and Winston, New York, 1987.
52. Ware, W. H. The Ultimate Computer, IEEE Spectrum, Vol. 9, No. 3, pp. 84-91, 1972.
53. Flatt, H. and Kennedy, K. Performance of Parallel Processors, Parallel Computing (to appear).
54. Gustafson, J. L., Montry, G. R., and Benner, R. E., Development of Parallel Methods for a 1024-Processor Hypercube, SIAM Journal of Scientific and Statistical Computing, Vol. 9, No. 4, pp 609-638, July 1988.
55. Gustafson, J. L., Reevaluating Amdahl's Law, Communications of the ACM, Vol. 31, No. 4, pp. 532-533, 1988.
56. Rodrigue, G. Some Ideas for Decomposing the Domain of Elliptic Partial Differential Equations in the Schwarz Process, Communications in Applied Numerical Methods, Vol. 2, pp. 245-249, 1986.
57. Carey, G. F. Parallelism in Finite Element Modeling, Communications in Applied Numerical Methods, Vol. 2, pp. 281-287, 1986.
58. Flower, J., Otto, S. and Salama, M. Optimal Mapping of Irregular Finite Element Domains to Parallel Processors, in Parallel Computations and Their Impact on Mechanics (Ed. A. K. Noor), pp. 239-250, in Proceedings of the Symposium on Parallel Computations and Their Impact on Mechanics, Boston, MA, Dec. 1987, AMD Vol. 86, American Society of Mechanical Engineers, New York, 1987.
59. Farhat, C. H. and Wilson, E. L. A New Finite Element Concurrent Computer Program Architecture, International Journal for Numerical Methods in Engineering, Vol. 24, pp. 1771-1792, 1987.
60. Nour-Omid, B. and Park, K. C. Solving Structural Mechanics Problems on the CALTECH Hypercube Machine, Computer Methods in Applied Mechanics and Engineering, Vol. 61, No. 2, pp. 161-176, 1987.
61. Irons, B. M. A Frontal Solution Program for Finite Element Analysis, International Journal for Numerical Methods in Engineering, Vol. 2, pp. 5-32, 1970.
62. Hayes, L. J. Advances and Trends in Element-by-Element Techniques, in State-of-the-Art Surveys on Computational Mechanics (Ed. A. K. Noor and J. T. Oden), American Society of Mechanical Engineers, New York (to appear).
63. Noor, A. K. and Whitworth, S. L. Computational Strategy for Analysis of Quasi-symmetric Structures, Journal of Engineering Mechanics, ASCE, Vol. 114, No. 3, pp. 456-477, 1988.
64. Noor, A. K. and Whitworth, S. L. Vibration Analysis of Quasi-symmetric

ORIGINAL PAGE IS
OF POOR QUALITY

Structures, Finite Elements in Analysis and Design, Vol. 3, No. 4, pp. 257-276, 1987.

65. Noor, A. K. and Peters, J. M. Model-Size Reduction for the Nonlinear Dynamic Analysis of Quasi-symmetric Structures, Engineering Computations, Vol. 4, pp. 178-189, 1987.

66. Noor, A. K. and Peters, J. M. A Partitioning Strategy for Efficient Nonlinear Finite Element Dynamic Analysis on Multiprocessor computers, Computers and Structures (to appear).

Supercomputing In Engineering Structures

Editors:
P. Melli
C.A. Brebbia

Computational Mechanics Publications
Southampton Boston

Co-published with

Springer-Verlag Berlin Heidelberg New York
London Paris Tokyo

P.MELLI.
IBM ECSEC.
Via Georgione, 159
00144
Roma
Italia

C.A. BREBBIA
Computational Mechanics Institute
Ashurst Lodge
Ashurst, Southampton
SO4 2AA
UK

British Library Cataloguing in Publication Data

Supercomputing in engineering structures
1. Structures. Analysis. Applications of supercomputer systems
I. Melli, Piero II. Brebbia, C.A. (Carlos Alberto), 1938-
624.1'71'0285411
ISBN 1-85312-020-0

Library of Congress Catalog card number 88-63525

ISBN 1-85312-020-0 Computational Mechanics Publications Southampton
ISBN 0-945824-07-6 Computational Mechanics Publications Boston, USA
ISBN 3-540-50687-X Springer-Verlag Berlin Heidelberg New York London Paris Tokyo
ISBN 0-387-50687-X Springer-Verlag New York Heidelberg Berlin London Paris Tokyo

Printed in Great Britain by The Eastern Press Limited, Reading

CONTENTS