Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

FUTURE REAL-TIME TESTING AND EVALUATION COMPUTING ENVIRONMENTS

Douglas R. Goodman

Cray Research Inc.
894 Ross Drive, Suite 203
Sunnyvale, California 94089

Robert G. Enk

Cray Research Inc.
Suite 1331 North
1331 Pennsylvania Avenue, NW
Washington, DC 20004

ABSTRACT

Just as future aerospace programs will be more than simply bigger or faster versions of existing systems, so too must future computational systems be more than bigger or faster versions of existing capabilities. Future aerospace programs will be discriminated from current programs by their ability to operate as an integrated system-of-systems rather than as a loose collection-of-systems. Similarly, the computational environment required to support the analysis and operation of future aerospace programs cannot be derived as a simple extension to, or collection of, existing computational capabilities. This paper explores the requirements for an integrated, high performance, real-time computing environment capable of testing and evaluating future aerospace systems.

1. A THIRD METHODOLOGY

The evolution of the role of computational simulation in the design, analysis and operation of modern aerospace systems is causing an important transformation in the science of real-time testing and evaluation. This evolution can be best understood from an historical perspective that highlights both the importance of computational methods to the advancement of modern science, and its shortfalls in terms of its completeness as a mature methodology.

Figure 1 depicts the emerging view of computational science as a peer level methodology to traditional theoretical and experimental methodologies. This view challenges the conventional view that considers computational technologies as simply a support tool for other sciences. This emerging view states that computational sciences have introduced a third methodology for the design and analysis of modern systems.

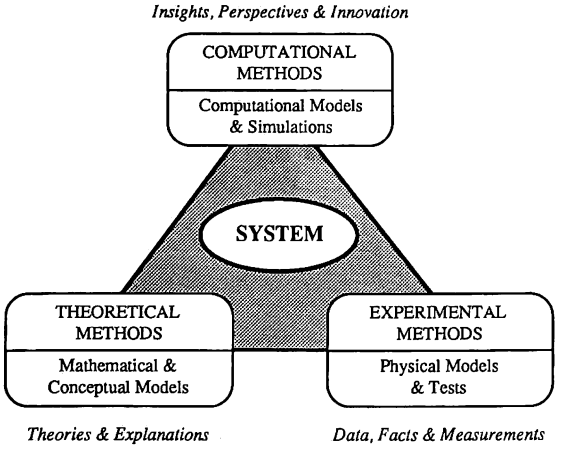

Figure 1. A Third Methodology

This perspective was first recognized within the Department of Energy's Nuclear Weapon Design program and was well documented in a report to Congress entitled "The Need for Supercomputers in the Nuclear Weapon Design Program" [Wilson et al. 1986]. In this document experimental and theoretical methodologies are described in terms of their relationships as well as their limitations. It first describes how experiments provide the real world data and measurements from which theoretical models are built; and how theoretical techniques are then used to develop predictive models that are validated by experimental testing programs. Both methodologies are, however, highly dependent upon direct human experience. Current theories are based on what we perceive to be logical and consistent with observations, and experiments can only test what they are designed to measure. Where theory is incomplete we make assumptions (particularly about relationships), and where physical testing is impractical we conduct representative tests.

As the systems we design and analyze begin to exceed direct human experience and human intuition, experimental and theoretical methodologies alone become inadequate. While this was first encountered in the practical application of quantum mechanics and relativistic physics to nuclear weapon design, it is true of any system that cannot be completely physically tested or for which a fully integrated theoretical model cannot be developed. Computational methods provide a means of eliminating the assumptions inherent within theoretical models and expanding the testing environment beyond that which is physically practical. Computational methods complement traditional methodologies by providing insights into the operation of complex systems that would otherwise be overlooked by theoretical methods or left untested by experimental methods. These new insights are the source for not only better designs but also the source for the identification of failure modes that would have otherwise gone undetected until they are experienced under real operational conditions.

Exploiting this emerging relationship between experimental and computational methodologies offers significant potential in the testing and evaluation of future aerospace systems. This work is dependent upon our ability to provide an integrated computing environment that integrates these methodologies. Some examples of how these relationships are being exploited today will serve as a basis for extrapolating future requirements and directions.

1.1 Nuclear Weapon Design Example

The Nuclear Weapon Design process is made exceptionally complicated by the difficulties involved in physical testing. Physical, economic, and political restrictions severely limit nuclear weapons testing. As a result, nuclear weapon designers depend heavily upon supercomputing as a platform for conducting extensive computational tests. The capabilities of the computational systems define the limits of practical nuclear weapon designs, as shown in Figure 2.

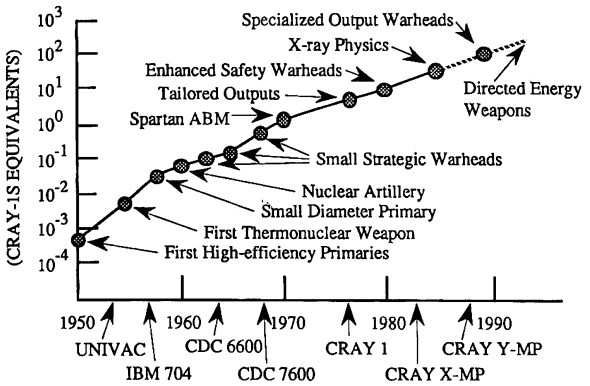

Figure 2. Supercomputing Use In Nuclear Weapon Design

In most design efforts of any complexity, experimental facilities can be set up to obtain data on the device before actually building a full scale prototype. Wind tunnels, for example, are used in the aerospace industry. There is no parallel to the wind tunnel in nuclear weapons design; no experimental facility exists which can pretest critical nuclear design parameters. Processes which only occur during a nuclear explosion itself cannot be evaluated. In addition the physical tests that are conducted are limited in scale (less than 150 kilotons) and in environments (underground) such that the performance of actual devices can only be inferred.

Therefore in Nuclear Weapon Design:

- **Computational Methods are the Experimental Environment** - The limits to which computational experiments can be conducted define the limits of practical designs.
- **Physical Tests Support Computational Experiments** - Nuclear tests are specifically constructed to provide data to validate and evaluate computational experiments.

1.2 Crash Testing Example

Computational crash testing is used in aerospace and automotive industries to optimize designs for structural integrity under complex impact conditions. These techniques were originally developed to test designs under conditions that could not be physically recreated (hypervelocity impacts), but have since been extended to provide broader capabilities.

Physical tests are limited for two reasons. First, complete instrumentation is difficult to achieve. Most physical tests, therefore, provide limited data on the intermediate effects and often only provide before and after results. Secondly, destructive testing often introduces dust, smoke or other artifacts that obscure the testing process, making techniques such as high speed photography ineffective in capturing data.

Computational models are not subject to these limitations. They can be instrumented at virtually any spatial or temporal resolution desired by the tester to provide complete detail of the intermediate dynamics. And the artifacts of the testing process can be masked to provide direct evaluation of the parameters of interest.

Therefore, in crash testing:

- **Computational Methods Provide Intermediate Detail** - They provide data that cannot be obtained with before and after destructive testing methods.
- **Computational Methods Eliminate Physical Testing Artifacts** - They allow removal of unwanted information that would otherwise obscure the experimental data collection and analysis process.

As an example of computational crash testing, Figure 3a depicts the results of a 50 kilometer per hour frontal impact test. The large finite element model used to obtain results for further vehicle development is shown [Nalepa et al. 1987]. The model consisted of 7911 shell elements and 7233 nodes.

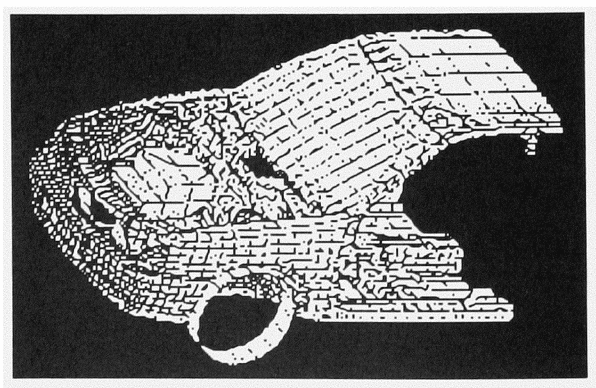

Figure 3a. Finite Element Model

An accurate description of the location of the center of gravity, masses, and mass inertia was also provided. Figure 3b shows the deformation of the vehicle 30 milliseconds after impact.

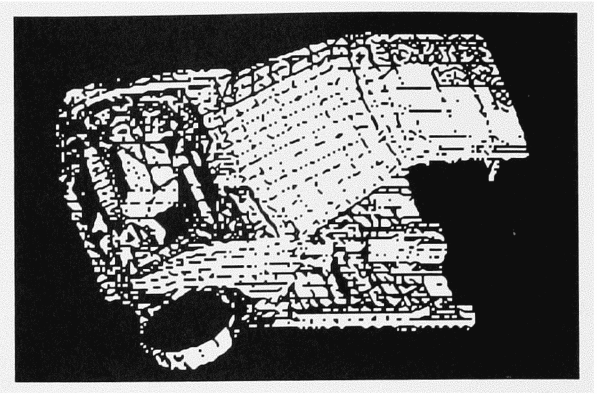

Figure 3b. Deformation After Crash

Among other information, the simulation showed that the siderails greatly influence vehicle crash behavior, because more than 40% of the kinetic energy of a frontal impact will be absorbed by the siderails. Global variables such as forces, velocities and deformations can also be defined, enabling the development engineer to evaluate crash behavior of different structural modifications at an early stage.

Figure 4 shows the comparison between the experiments done and the computer simulation, for the velocity and deformation value changes in the 50 kilometer-per-hour frontal impact.

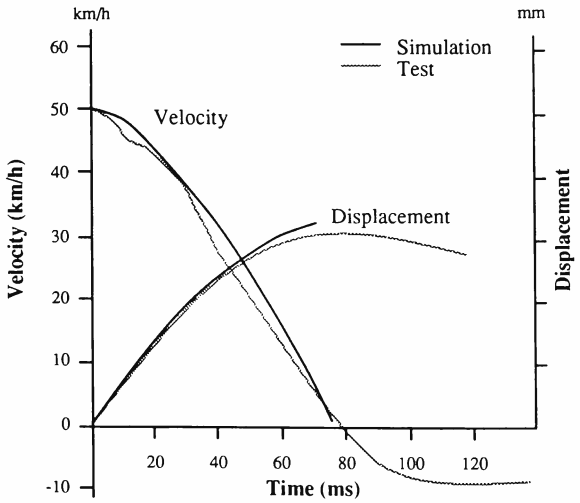

Figure 4. Comparison of Test and Analyses

Note that for accuracy reasons (round-off error accumulation) only machines such as Cray Research computer systems with at least 64-bit single precision are suitable for running crash simulation programs. Large, high performance, real memories and high storage capacity are also necessary .

1.3 Computational Fluid Dynamics Example

The science of Computational Fluid Dynamics (CFD) has long been the "showcase" application for computational methods [Peterson 1984, 1989]. As shown in Figure 5a, CFD and supercomputing technologies have evolved hand-in-hand since the mid 1970's when the supercomputer of that day, the CDC-7600, was used to computationally simulate the two dimensional airflow around the cord of an aircraft wing. By 1983 the CRAY-1 was capable of simulating the three dimensional airflow around an entire wing (Figure 5b), and by 1986 entire simplified aircraft structures were being simulated by the CRAY X-MP/4 (Figure 5c). During this period computational models have been primarily used to sup-

port wind tunnel testing programs or to explore environments outside the performance limits of the wind tunnels. Recent advancements, however, have resulted in a new relationship between computational and experimental methods. With the availability of CRAY Y-MP class supercomputers it is possible to not only model complete aerospace structures, but to model them in a simulated wind tunnel environment (Fig 5d). Computational wind tunnel experiments are now sufficiently accurate that the anomalies introduced by physical wind tunnels, such as wall and support interference can be isolated.

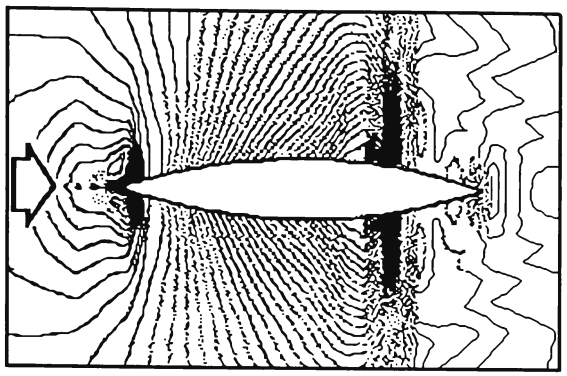

Figure 5a. 1975 CDC-7600

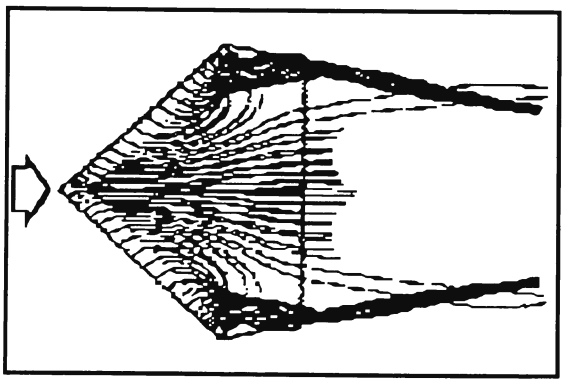

Figure 5b. 1983 CRAY-1

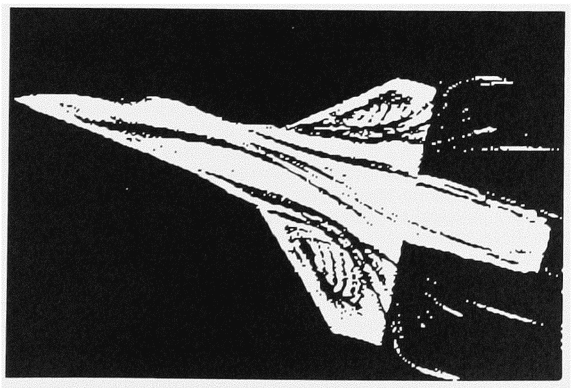

Figure 5c. 1986 CRAY X-MP/4

Therefore, in CFD testing:

- **Computational Methods correct Physical Tests** - They isolate and subtract out the anomalies introduced by physical testing systems.
- **Computational Methods Account for Effects** - They account for phenomena that cannot be physically simulated.

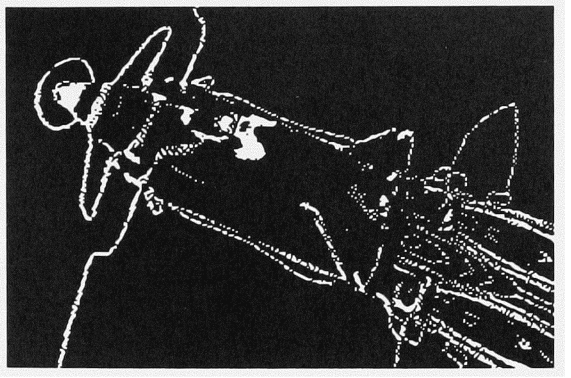

Figure 5d. 1989 CRAY Y-MP8/8

2. COMPUTATIONAL LIMITATIONS

Today's high performance computing architectures have evolved along three distinct paths in direct response to the computational requirements of the three methodologies described earlier. Figure 6 depicts these three environments in terms of the principle factors that have influenced their basic architectures. High performance scientific computers (supercomputers) have been optimized to support large scale, high fidelity scientific processing. High performance operational computers (real-time systems) have been optimized to support synchronization and interaction with external devices and systems. And, high performance system analysis computers (studies and ops-analysis systems) have been optimized to support large scale data base management and correlation functions. Unfortunately, the process of focusing on the requirements of one methodology has resulted in computational incompatibilities between the methodologies and sub-optimal solutions for integrating capabilities between the methodologies.

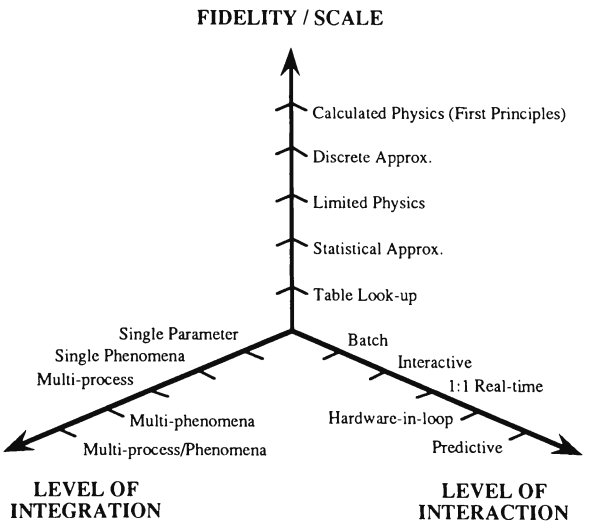

Figure 6. Three Computing Environments

Supercomputers (in traditional configurations) have not been specifically engineered to support real-time levels of interaction or used for complex operational analysis of multi-process or multi-phenomena systems.

Real-time computers are primarily designed to support high performance I/O, and are typically not robust enough computationally to perform complex calculations in small frame times. Often these complex calculations are done in advance and results are stored as look-up tables that are accessible by the real-time applica-

tion during real-time. Where multiple simultaneous applications must be supported, real-time systems are often configured as distributed processors with one processor dedicated to each real-time job. Process-to-process coordination must then be managed through networks.

Large scale operations analyses, such as strategic planning,, generally rely on large data base computers that use parametric models and statistical methods to describe complex system interactions. These machines typically are limited in their scientific processing capabilities. Complex interactions are often simplified to probabilities that lend themselves to data base techniques. The slow performance of data base technology limits these systems' usefulness in real-time, operational environments.

3. INTEGRATED SIMULATION FRAMEWORKS

The computational systems required to support a future design, analysis and operational environment must be a synthesis of all three of these computing environments. The challenge of next generation computing systems is to integrate these discrete environments into a high performance, tightly-coupled framework.

Such an Integrated Simulation Framework (ISF) must be capable of simultaneously supporting 1) high fidelity multi-phenomena analysis; 2) real-time, Hardware-in-the-Loop interfaces; and 3) large-scale multi-process parametric modeling.

Such frameworks do not exist today. However, conceptual frameworks and initial designs have been explored. One of the more aggressive efforts was developed as part of the DETEC program to support SDI National Test Bed requirements [Christman et al. 1986]. While it is beyond the scope of this paper to describe DETEC or represent its current functional capabilities at the National Test Bed, the underlying concepts are instructive.

The ISF postulated by the DETEC project was composed of the four major functional subsystems shown in Figure 7.

1) The Simulation Space Function provides a computational environment within which all the physical phenomena models could co-exist in an internally consistent format. The function also provides a mechanism for establishing and simulating the logical and physical interfaces between all the simulated and real elements of the system.

2) The Object Simulation Function allows the explicit definition and self contained simulation of each independent element in the system including all its non-deterministic attributes.

3) The Simulator /Interface Function allows the interfacing of externally simulated or real-world devices into the overall simulation in a format identical to the internally simulated elements.

4) The System Management Function provides for the overall control of the system as a discrete event simulation capable of integrating and synchronizing real and simulated elements in a large-scale system-of-system framework.

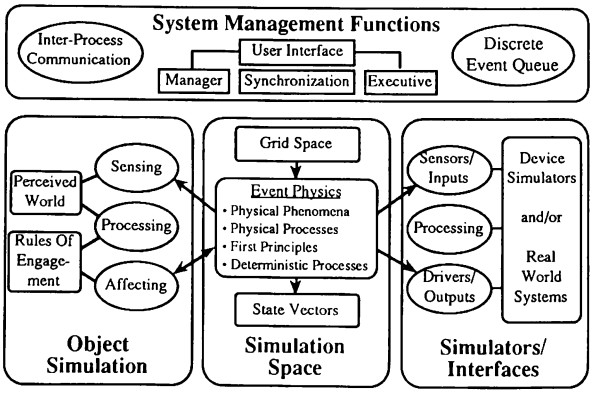

Figure 7. Integrated Simulation Framework

3.1 Simulation Space Function

The Simulation Space Function is composed of three major sub-functions; Grid Space Functions, State Vector Functions and Event Physics Functions.

The Grid Space is the spatial definition of the physical environment within which the simulation will occur. This function maintains the current values of all the first principle parameters (temperature, pressure, etc.) relevant to each grid point. The interface and coupling between multiple phenomena is established and maintained through the Grid Space Function. That is, the initial conditions for each iteration of a process are drawn from the same first principle parameter set, and the resulting effects are shared between the processes and are reflected as grid quantity updates and/or changes to state vectors.

The State Vector is the explicit definition of each independent object (simulated and real) within the simulation space. In addition to the traditional position, velocity and attitude state vectors are all the explicitly definable values of each object such as health/status conditions, geometric characteristics and intrinsic values such as optical and radiation hardness.

Event Physics is the collection of all the physical phenomena models that support the modeling of the Simulation Space, such as atmospheric effects and terrain dynamics, and also all the physical processes of the simulated and real objects, such as booster burn physics and weapons effects physics. Event Physics is structured as a collection of validated first principle algorithms not as one large integrated code.

3.2 Object Simulation Function

The Object Simulation Function provides a uniform framework for the simulation of the non-deterministic attributes of all the discrete objects (aircraft, command centers, missiles, etc.) that operate within the simulation space. Each object in the system is separately identified and characterized in terms of its ability to sense or *input* information from the simulation space, analyze or *process* that information, and to affect or *output* back into the simulation space.

To characterize a real object, such as a missile system, a hierarchy of these I/P/O constructs may be necessary to define all the relevant subsystems that compose the object. In addition to these I/P/O modules, each object maintains a unique perspective of the Simulation Space called its Perceived World that is built up from the information accumulated through the sensing module including all the misperceptions resulting from communications and sensor errors.

The Rules of Engagement function provides each object with a set of directives for initiating actions based on its perceptions of the state of the local environment. The Affecter module is implemented as a parametric representation of the physical characteristics of the relevant outputs such as thrust profile for a missile or frequency and power output for a transmitter. The actual execution of the resulting physical processes and the consequential effects on the local environment and interaction with other objects is handled by the Event Physics modules in the Simulation Space Function. Therefore, there are no explicitly defined interactions between objects in the system; all interactions (intended and unintended) are implicit to the simulation process.

3.3 Simulator/Interface Function

The Simulator/Interface Function provides a framework for the interfacing of externally simulated or real-world devices into the overall simulation in a format compatible with the Object Simulator Function. The Sensor module allows simulated sensor information from the Simulation Space to be interfaced to external devices that are either physical simulators or actual systems that can be driven from these simulated inputs. The Driver module allows outputs from external physical simulators or actual systems to be interfaced into the Simulation Space. This module also allows actual parameters to be introduced into the Grid Space and State Vector Space as real-world values in lieu of computationally derived values.

3.4 System Management Function

The System Management Function provides for the overall control of the system as a discrete event simulation synchronized with real-world time scales. The computing elements of the system must, therefore, be sized to support this event driven processing and the scheduling functions must be sufficiently predictive to provide the necessary synchronization of outputs. The System Management Function provides the resource management and scheduling of processes to maintain this synchronization as well as the interprocess

communications required to integrate the various functions.

3.5 ISF Benefits

The overall ISF is exercised through a series of cause and effect event threads, as shown in Figure 8, that are initiated by internally simulated deterministic processes within the Event Physics module, by simulated non-deterministic processes within the Rules of Engagement modules, and by real-world events and external conditions stimulated from the Simulator/Interface Function.

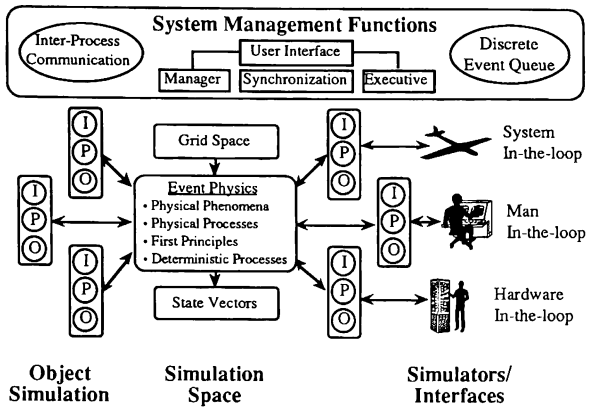

Figure 8. Integrated Simulation Environment

An Integrated Simulation Framework offers the following unique advantages over traditional, non-integrated computational support systems:

- **It provides an implicit (versus explicit) modeling framework for analyses.** Models that depend on explicit definitions for all the possible inter-relationships between the elements and processes of complex system-of-system simulations have two drawbacks. They are logically limited by their inability to predict all the intended and untended interrelationships, and they are computationally limited by their inability to support the complex network of interprocess communications required to support each explicit relationship.
- **It provides a mixed mode of digital simulation and physical simulators.** By providing an environment that allows digital models to interact with real-world systems, theoretical models can be directly and interactively compared with real-world conditions, and real-world systems can be tested in more complete system-of-systems environments.
- **It provides real-time access to large-scale scientific processing capabilities.** Making first principle analytical capabilities available to real-time processes minimizes the assumptions that must be made with statistical or data base techniques, and ensures a consistent application of physical processes across all elements of complex system-of-system simulations.
- **It provides a Life-Cycle support base for the design, analysis and evaluation of complex aerospace systems.** During the early design phases the ISF is structured as an all digital simulation. During the development phases the ISF supports the early analysis of subsystems and prototype elements as a mixture of digitally simulated and hardware-in-the-loop components. As part of the OT&E of the system, the ISF can then evolve to a simulator platform for full scale system-of-system evaluations.

4. SUPERCOMPUTING AND INTEGRATED SIMULATION FRAMEWORKS

Supercomputing technologies will play a critical role in developing Integrated Simulation Frameworks. While supercomputers are most commonly associated with their ability to deliver very high levels of performance on large scientific applications, the underlying technology is much broader than just raw CPU performance. Modern supercomputer technologies, typified by Cray Research products, represent a balance of high performance processing, I/O, memory, networking and software technologies. What distinguishes supercomputing from other forms of high performance computing systems is a balanced architecture that can deliver this performance across a range of applications and user environments.

Of specific importance to the development of Integrated Simulation Frameworks are both the internal parallel/vector architecture, and the very high performance internal and external I/O capabilities. The parallel/vector architecture of multiple CPUs with scalar and vector functional units is uniquely suited to supporting tightly coupled multi-phenomena analysis. The very high performance I/O structure is an ideal platform for implementing a very high performance real-time, hardware-in-the-loop capability that can be tightly coupled with the high performance processing resources.

Cray Research supercomputer hardware and real-time software enhancements provide several orders of magnitude more processing power than is presently available for real-time applications. This processing power can be applied to real-time simulation applications, supporting submillisecond real-time frame times.

Figure 9 provides an overview of a typical Cray Research Y-MP system, which is described below.

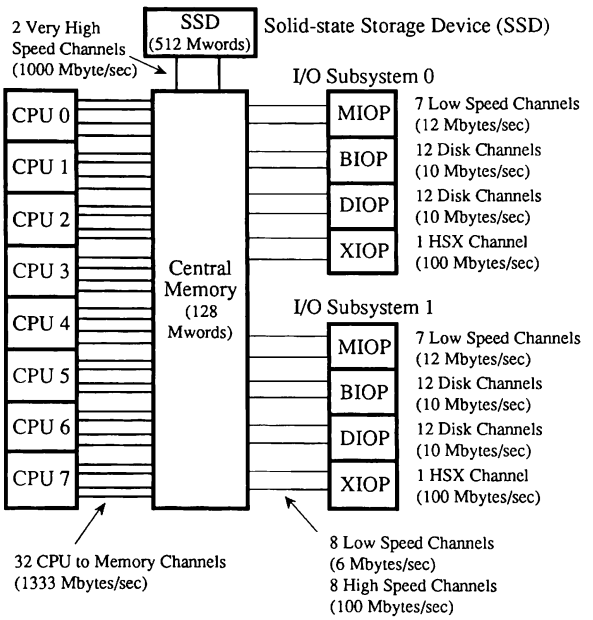

Figure 9. Cray Research CRAY Y-MP8/8128 Architecture

The CRAY Y-MP8/8128 is an eight-processor, 128 Mword (64-bit word) supercomputer with a 6 nanosecond clock. The high performance of the eight CPUs is complemented by the very high memory bandwidth, high speed solid-state storage bandwidth, and broad connectability to the external world. The 8 processors of the CRAY Y-MP8 share a common central memory, and can simultaneously process separate programs or, under user control, different portions of a single program. The CRAY Y-MP8 system includes one or two high performance I/O Subsystems (IOS) that provide up to 14 low speed (6 or 12 Mbyte/second) external channels, one or two high speed (100 Mbyte/second) external channels, and the capacity to support up to 48 independent disk channels, each rated at 10 Mbyte/second each. Total disk storage supported is 480 Gbytes. A solid-state storage device (SSD), accessed from the CPU over one or two 1000 Mbyte/second channels, is available with up to 512 Mwords of storage.

Each CRAY Y-MP8 processor is theoretically capable of providing 333 Mflops for a total theoretical performance of 2.667 Gflops. On kernel codes, the CRAY Y-MP8 has demonstrated 2.144 Gflops on the 1000x1000 LINPACK kernel, and 2.36 Gflops on a matrix multiply kernel. Real CFD codes such as ARC3D, a

three dimensional CFD code, has been demonstrated to execute at 1.134 Gflops, and THRED, another CFD code, at 1.397 Gflops. A 128 code aerospace job mix was demonstrated to run at a sustained 1.156 Gflops.

CRAY Y-MP8 memory bandwidth is equally impressive. Each CPU communicates to memory over four ports, two write and one read, plus an I/O port, each rated at 1333 Mbytes per second. If a situation were to occur wherein all these ports were active, total memory bandwidth would be 42.6 Gbytes/second.

The optional SSD provides a very fast auxiliary memory which can be used for I/O caching to disk devices, user or system file space, or as shared memory between processes. Actual data transfer rates of 1.8 Gbytes per second (using two 1 Gbyte/second channels) have been demonstrated to and from this device.

The one or two IOS subsystems provide the interface between the CRAY Y-MP8 mainframe and the external world. The 14 available low speed (6 Mbyte or 12 Mbyte/second) channels can be connected to medium speed networks such as NSC HYPERchannel, or to VME-based devices such as Sun or Motorola workstations. These workstations or IP routers such as the NSC EN643 can be used to connect to Ethernet, FDDI, or other low speed networks. In addition, a number of direct point-to-point links to other vendor computers are supported over these channels, including DEC, IBM, CDC, and others.

The two available high speed (100 Mbytes/second) channels can be connected to UltraNetwork Technologies' UltraNet network, to HiPPI, or to special customer-furnished devices.

With high performance CPUs, memory bandwidth, extensive high performance I/O capability, and large memories and disk capacity, the CRAY Y-MP8 demonstrates the balanced architecture required to to allow the integration of multiple computational capabilities to support the integrated simulation frameworks needed for future aerospace systems.

Cray Research's UNICOS operating system supports the balanced architecture described above. Based on AT&T's System V UNIX, UNICOS provides interactive, local batch, and remote batch user interfaces. Among many other enhancements for supercomputer users, the TCP/IP and ISO networking standards are supported. This networking support allows Cray Research supercomputers to provide extensive capability for scientific visualization to users.

The Cray Research Multi-Purpose Graphics System (MPGS) is designed to take full advantage of the Cray Research supercomputer running UNICOS and a high-performance Silicon Graphics workstation in a distributed environment. The workstation runs the easy-to-use user interface, and performs local transformations by use of workstation hardware. Processor and memory intensive tasks are distributed to the Cray Research supercomputer. MPGS performs and creates transformations, hidden surfaces, hidden lines, contouring, vectors, particle traces, clipping and a false color map.

Use of such graphics capabilities greatly enhances the ability of users to understand and expand the complex interactions of physical phenomena within simulation work. Such understanding is vital to the continued development of complex simulations within the Integrated Simulation Framework described above.

5. SUMMARY

The effective testing and evaluation of future aerospace systems will require the integration of theoretical, experimental and computational methodologies. Integrating these methodologies will require the development of an Integrated Simulation Framework that couples the computational capabilities that have evolved independently of one another to support these disciplines. The balanced architectures inherent within current and future supercomputing technologies can provide a high performance platform for facilitating this integration.

REFERENCES

Christman, R.D., Dana, A.L., Henderson, D.B., Shafer, B.P., Wood, J.A., and Wood, M.M. (1986), "Conceptual Specification for Defensive Technology Evaluation Code (DETEC)", Technical Report LA-10547-MS, Los Alamos National Laboratory, Los Alamos, NM.

Nalepa, E.J., and Le-The, H. (1987), "Crashworthiness Simulation: An Emerging Tool for Vehicle Design Optimization, *Cray Channels*, 8, 4, 8-11.

Peterson, V. (1984), "Impact of Computers on Aerodynamics Research and Development", *Proceeding of the IEEE* 72, 1, 68-79.

Peterson, V. (1989), "Supercomputing: A Way to Leadership in Aerospace", a briefing, NASA Ames Research Center, Mountain View, CA.

Wilson, W.D., Gallagher, R., et al. (1986), "The Need for Supercomputers in Nuclear Weapon Design", Report to Congress, U.S. Department of Energy, Los Alamos National Laboratory, Los Alamos, NM, Lawrence Livermore National Laboratory, Livermore, CA, Sandia National Laboratories, Sandia, NM.