

EVALUATION OF THE INTEL IWARP PARALLEL PROCESSOR FOR SPACE FLIGHT APPLICATIONS

Butler P. Hine III
Information Sciences Division
NASA Ames Research Center
Moffett Field, CA 94035

Terrence W. Fong
Recom Technologies, Inc.
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

The computer resources on board Space Station Freedom (SSF) are in high demand due to design constraints on weight, power, and volume. To achieve acceptable performance levels for station operation as well as computationally stressing applications, it is important to critique computer system architectures for their ability to meet computational requirements. In particular, this research seeks to evaluate the potential of a DARPA-sponsored advanced processor, the Intel iWarp, for use in future SSF Data Management Systems (DMS) upgrades through integration into the Ames DMS testbed and applications testing. The iWarp is a distributed, parallel computing system well suited for high performance computing applications such as matrix operations and image processing. The system architecture is modular, supports systolic and message-based computation, and is capable of providing massive computational power in a low-cost, low-power package. As a consequence, the iWarp offers significant potential for advanced space-based computing. This research seeks to determine the iWarp's suitability as a processing device for space missions. In particular, the project focuses on evaluating the ease of integrating the iWarp into the SSF DMS baseline architecture and the iWarp's ability to support computationally stressing applications representative of Space Station Freedom tasks.

Introduction

The computer resources on board Space Station Freedom (SSF) are in high demand due to design constraints on weight, power, and volume. To achieve acceptable performance levels for station operation as well as computationally stressing applications, it is important to critique computer system architectures for their ability to meet computational requirements. Parallel processing systems show promise in achieving the computational speed required for modeling physical phenomena for space operations¹, but within the tight constraints on power consumption, weight, and volume. In particular, this research seeks to evaluate the potential of a DARPA-sponsored advanced processor, the Intel iWarp Processor,

for use in future SSF Data Management System (DMS) upgrades through integration into the Ames DMS testbed and through applications testing².

The iWarp

The iWarp is a distributed, parallel computing system well suited for high performance computing applications such as matrix operations and image processing³. The system architecture is modular, supports systolic and message-based computation, and is capable of providing massive computational power in a low-cost, low-power package. As a consequence, the iWarp offers significant potential for advanced space-borne computing.

The design objectives for the iWarp project were to provide a balanced (1:1) communication to computation ratio, to support high performance image and signal processing applications, and to provide a modular, scaleable system with high-level programming tools. The result of the project was the iWarp, a distributed memory, multiprocessing multiple instruction, multiple data stream (MIMD) computer system. Each processor can operate autonomously (loosely coupled computing) in a coarse-grained message-based communications manner similar to the iPSC/860 (Hypercube), or cooperatively (tightly coupled computing) in a fine-grained systolic communications manner. Some interesting features of the architecture are: logical channels (for multiplexed connections on single physical link), systolic gates (for direct computation access to incoming data), and a long instruction word on a otherwise RISC-like processor. The iWarp component specifications are as follows:

- RISC-like processor
- 650,000 transistors
- Independent computation and communication
- 24-bit address and 64-bit data bus
- 32-bit and 96-bit (LIW) instructions
- 20 MIPS and 20 MFLOPS (single-precision) with 20 MHz clock
- 320 MB/sec communication throughput per cell
- 160 MB/sec data transfer to local memory
- 40 MB/sec, full-duplex, data transfer between cells

Figure 1 shows a schematic diagram of the iWarp parallel processor. The system in use at the Ames Research Center (ARC) consists of 8 cells in a 2D toroidal array, with 512 KBytes per cell. The interface to the Sun-4 host is through the Sun Interface Board (SIB). Note that each cell has both computation and communication agents. The hardware topology is physically a 2-D toroidal mesh, with each cell in the array connected to four neighboring cells (x-left, x-right, y-up, y-down). The logical connections, however, support arbitrary connectivity (i.e., software programmable network topologies). The input/output (I/O) for the array is handled by special "interface cells" (an iWarp cell augmented with dual-ported RAM). The only interfaces that currently exist are the SIB, which allows communication between a Sun workstation and the array and a serial interface board. The latter provides 60 MB/sec transfer rates and is intended as a general purpose interface to high-speed devices (e.g. frame grabbers, video, disk).

The Problem

The research program at ARC seeks to determine the iWarp's suitability as a processing device for space missions. In particular, the project focuses on evaluating the ease of integrating the iWarp into the SSF DMS

baseline architecture and the iWarp's ability to support computationally stressing applications representative of Space Station Freedom tasks. The sample task we have chosen as representative of a purely computational load, as opposed to sensor processing, is that of calculating in "real-time" the dynamics of multiple interacting objects in low Earth orbit.

The environment around Space Station Freedom will at times include a number of co-orbiting vehicles and other objects. It is highly unlikely that all of these vehicles will remain in a stationary position with respect to the space station, and will therefore require a traffic management system to track and resolve conflicts⁴. This is an unusual situation, in comparison with past missions, in that it is highly dynamic, and cannot be completely precalculated on the ground. Since the orbital mechanics involved with multiple co-orbiting objects are complex and non-intuitive, a computational system capable of calculating the orbital dynamics in real-time is desirable. In addition, the dynamic time scales involved and the reliability of communication links make it preferable to have the traffic management system on-board the station, rather than on the ground.

Intel iWarp Parallel Processor

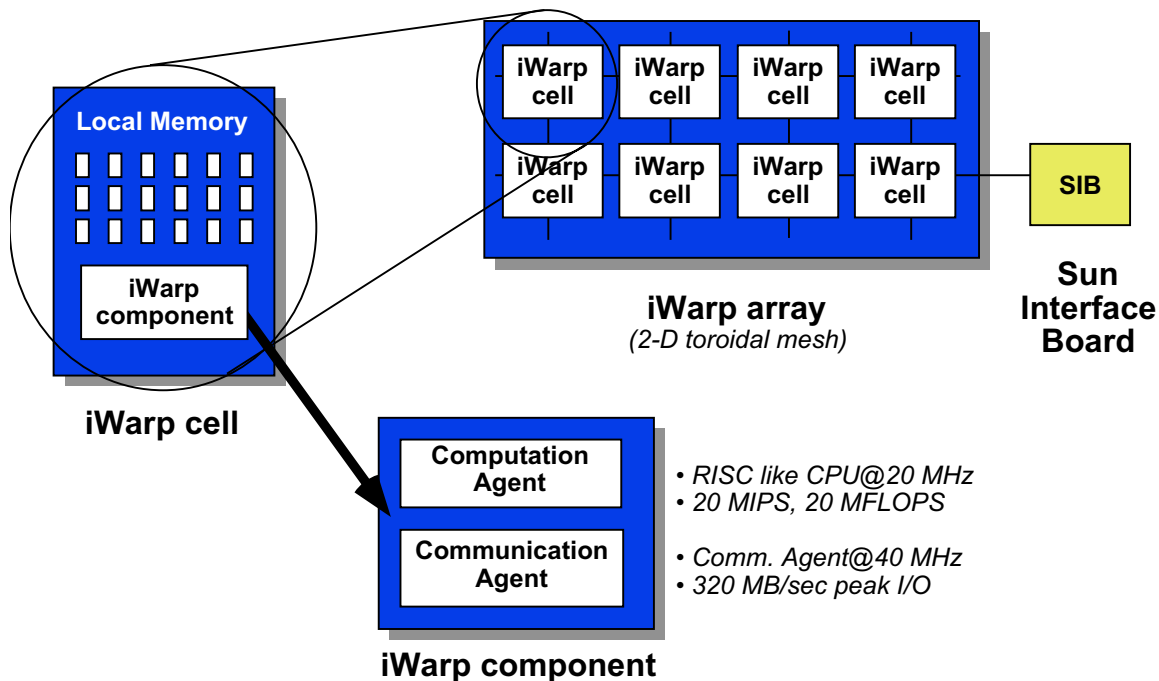


Figure 1: Schematic diagram of the Intel iWarp Parallel Processor. The ARC system consists of 8 cells in a 2D toroidal array, with 512 KBytes per cell. The interface to the Sun-4 host is through the Sun Interface Board (SIB). Note that each cell has both computation and communication agents.

The Algorithm

The specific problem chosen, simulating the orbital dynamics of multiple co-orbiting bodies, requires the solution of the equations of motion for each of the co-orbiting bodies, in the local vertical, local horizontal (LVLH) reference frame of the space station. The x-axis is along the radial direction from the Earth and the y-axis is along the the orbital path. The z-axis is normal to the orbital plane. The equations of motion are:

$$\ddot{\rho} = \frac{\mu}{r_1^3} \left[\vec{r}_1 - \frac{r_1^3}{r_2^3} \vec{r}_2 \right] + \vec{f}$$

These equations can be reduced, after Kaplan⁵ (see also Taff⁶) to *Hill's equations* :

$$\begin{aligned} \ddot{x} - 2\dot{y} - 3\dot{\theta}^2 x &= f_x \\ \ddot{y} + 2\dot{x} &= f_y \\ \ddot{z} + \dot{\theta}^2 z &= f_z \end{aligned}$$

The orbit of the station is assumed to be circular, at a height of 222 km. For the sample problem used in the timing tests, the initial conditions for an approaching shuttle on an intercept with the station are:

Orbit: 88.8 min. circular orbit at 222 km altitude
 Initial distance: x = 1000 m, y = 400 m
 Initial Velocity: vx = -3.14 m/sec, vy = -2.49 m/sec

We used a simple fourth order Runge-Kutta algorithm to integrate the equations of motion, with a step size of 1 msec.

iWarp Implementation

The iWarp component consists of separate communication and computation agents which can operate independently of the other, but since the agents are colocated, tight coupling may be achieved between communication and computation. Additionally, because the component is superscalar and a wide range of communications styles are supported, there is tremendous flexibility for implementing efficient, parallel algorithms. The iWarp is capable of supporting, for example, algorithms ranging from systolic matrix multiplication to coarse-grained, nonvectorizable, Monte Carlo simulations.

The iWarp communications agent offers a variety of services and communications styles. This flexibility allows the system to efficiently and gracefully support a wide range of parallel implementation methods. For example, the channel multiplexing capability of the iWarp component allows for a broad spectrum of connection topologies, including meshes, hypercubes and snakes. Additionally, the peak communications bandwidth of 320

MBytes/sec, low-latency transmissions, message passing service, and systolic communications support provides fine control over the communication and computation granularity⁷.

Data Parallel Algorithm

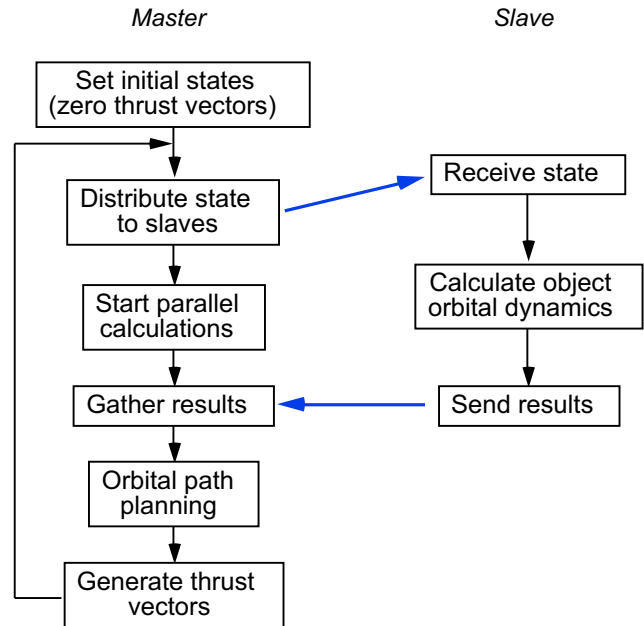


Figure 2: Top level flow chart of the orbital dynamics algorithm. Note that the master cell, the SIB, is responsible for controlling the overall flow of information. Each slave cell in the array is acting as a numerical integrator for individual objects.

The iWarp computation agent provides a powerful general-purpose processing engine. The superscalar processor is similar to other RISC systems, but can provide sustained computation rates for all codes through the use of a long instruction word (LIW). The LIW is extremely flexible and may be used to operate multiple functional units in parallel, to perform simultaneous computation and communication, and to provide optimizations such as zero-overhead loops and software pipelining. Most significantly, however, the computation agent was designed with an emphasis on sustained floating-point performance (20 MFLOPS). The processor achieves this computational rate through the use of nonpipelined floating-point units, and provides high performance for both vectorizable and non-vectorizable codes³.

For the initial implementation of the orbital dynamics algorithm, a coarse-grained message-passing model was

chosen for implementing task-level parallelism. This model is currently implemented using the "iWarp message-passing service" (imsg). This system was chosen after considering the two available iWarp communication libraries: imsg and "Programmed Communications Service" (PCS). PCS, which supports structured message-passing on static networks, provides higher communications bandwidth (39 MB/sec) than imsg (14 MB/sec) but is significantly more complex and requires greater development effort. Since the orbital dynamics problem was assumed to be compute-bound, the imsg bandwidth was considered to sufficient and ease-of-use favored imsg selection. Additionally, PCS requires the use of statically defined networks which severely limits ease of scalability. Finally, the imsg primitives strongly resemble message-passing primitives on other parallel systems (e.g., iPSC/860) and would allow the algorithm to be more easily ported to other environments.

The parallel implementation is shown in figures 2 and 3. A master node, the iWarp SIB, is responsible for controlling the overall flow of information through the simulation. Coarse-grained messages containing state data and control information is sent from the master node

to each of the slave nodes. Each slave node in the array performs numerical integration on orbital objects and returns the results to the master node. In the simplest configuration, each orbital object is tracked by a single node in the array. As the number of objects increases, the number of objects per node also increases.

It was discovered during development, however, that the orbital dynamics problem is not actually compute-bound but rather is communications-bound, and that the imsg bandwidth restricts overall performance. To improve performance of the parallel algorithm, it will be necessary to utilize PCS and a statically defined communications network. An example of this method is described in Fong⁸. In that application, a pair of unidirectional networks was used to perform "programmed scatter and gather" of data across the iWarp array. These networks maximize communications performance while utilizing a minimum of array resources. In particular, the ability of the communication agent on each node to forward data in an "express" mode (i.e., data is routed through the node without intermediate buffering) allows very low-latency, high bandwidth data distribution to be performed.

Parallel Processing Structure (data parallel decomposition)

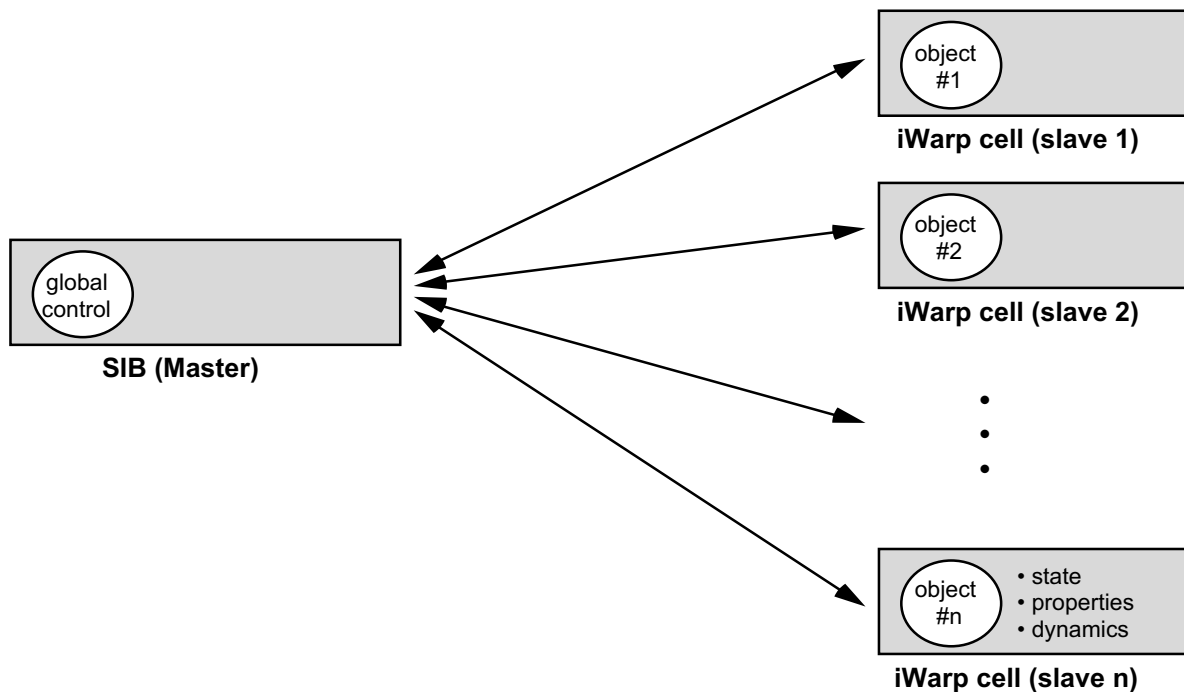


Figure 3: Schematic diagram of the computational load distribution for the orbital dynamics algorithm running in parallel on the iWarp array. In its simplest configuration, each orbital object is tracked by one cell of the array. As the number of objects increases, the number of objects per cell also increases. The SIB distributes commands and data to the cells of the array, and gathers the state vector information to be transmitted out of the array.

Visualization

To visualize the solution of multi-body orbital dynamics, the state data for each orbital object is broadcast at each time-step from the iWarp array to a Silicon Graphics (SGI) workstation. The SGI workstation uses this state data to update a real-time, three-dimensional (3D) graphical simulation of the orbital objects. The user may interact with the simulation to control the rendered viewpoint. A schematic diagram of the overall data flow is given below in Figure 4. A screen image of the 3D graphical simulator is shown in Figure 5.

The method used to broadcast the orbital object state data is the "Telerobotics Interconnection Protocol" (TelRIP), developed at Rice University and the NASA Johnson Space Center. TelRIP "is a data distribution mechanism designed for building complex distributed systems"⁹. It provides a data centered method for distributing data across standard networks using TCP/IP transmissions. Multiple processes, which may reside on different computers, register interest in specific data types. When such data becomes available, TelRIP immediately distributes the information to all processes which have

registered an interest. This approach allows information (e.g., orbital object state data) to be quickly and efficiently distributed.

Real-time rendering of the 3D graphical simulation is performed on a Silicon Graphics workstation using "World ToolKit" (WTK), developed by the Sense8 Corporation. WTK is a C library which provides functions for constructing real-time 3D graphical applications. The library is structured in an object-oriented fashion, is device independent, and runs on a variety of systems (i.e., Sun, SGI, IBM PC)¹⁰. Using WTK, the orbital dynamic simulation renders 22,000 polygons/second on an SGI Indigo Elan.

The combination of TelRIP and WTK has proven to be extremely efficient for visualization of the multi-body orbital dynamics. Since TelRIP is able to rapidly transmit data and because WTK can deliver high-performance rendering, the system provides a responsive visualization. Moreover, by allowing control of the rendered viewpoint in real-time, the graphical simulation is particularly effective for viewing spatial relationships.

Simulator Dataflow

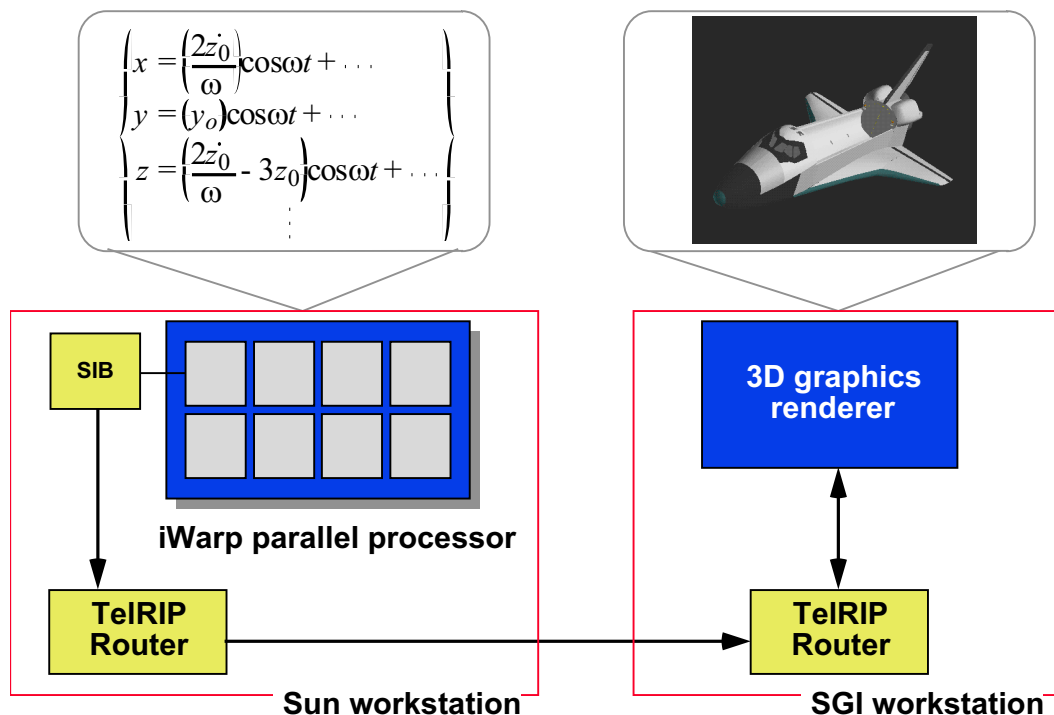


Figure 4: Schematic diagram of the algorithmic data flow for the orbital dynamics simulator. The dynamics equations are solved in real time on the iWarp array, and the state vectors for each orbital object are transmitted using the TelRIP communications protocol to a Silicon Graphics (SGI) workstation which performs the 3D rendering of the simulation. The user may interact with the simulation through the SGI, which relays commands to the iWarp array.

Performance

For the timing tests, we ran the same sample problem described above on the 8-cell iWarp and on its Sun-4/370 host computer. The timings below are for a 1 second integration of the equations of motion for 8 objects, using the fourth order Runge-Kutta integrator with 1000 steps. The execution time for the Sun-4 was 1.86 seconds, which is 0.24 seconds per orbiting object. The execution time for the iWarp for all 8 objects was 0.20 seconds, with 0.181 seconds individually per cell. Each cell was given one orbital body to compute. Each cell was therefore approximately 25% faster than the Sun-4 host, with the execution time for the whole problem approximately 9.3 times faster on the iWarp than on the Sun-4 host.

Conclusions

This application, while relatively small, represents a class of computational tasks for future space-borne processors.

Specifically, it encompasses many of the major issues faced by future high-performance, space qualified, parallel processors such as throughput, communications, command and control. Our goal is to try to solve space-borne processing tasks via parallel computing, and we believe that the best method to ascertain the effectiveness of such computing is through applications testing. The performance figures given above indicate that to achieve comparable performance from a single processor would require an order of magnitude greater clock rate, which in turn means that the single processor system would consume more power and dissipate more heat than for the parallel processor running at the slower clock rate. This demonstrates that for a class of compute bound problems amenable to simple parallelization, parallel computing can be very effective. The problems we have discovered specific to the iWarp, with respect to this application, mostly involve maturation of the product line. Our overall evaluation of the iWarp system to date is positive. The

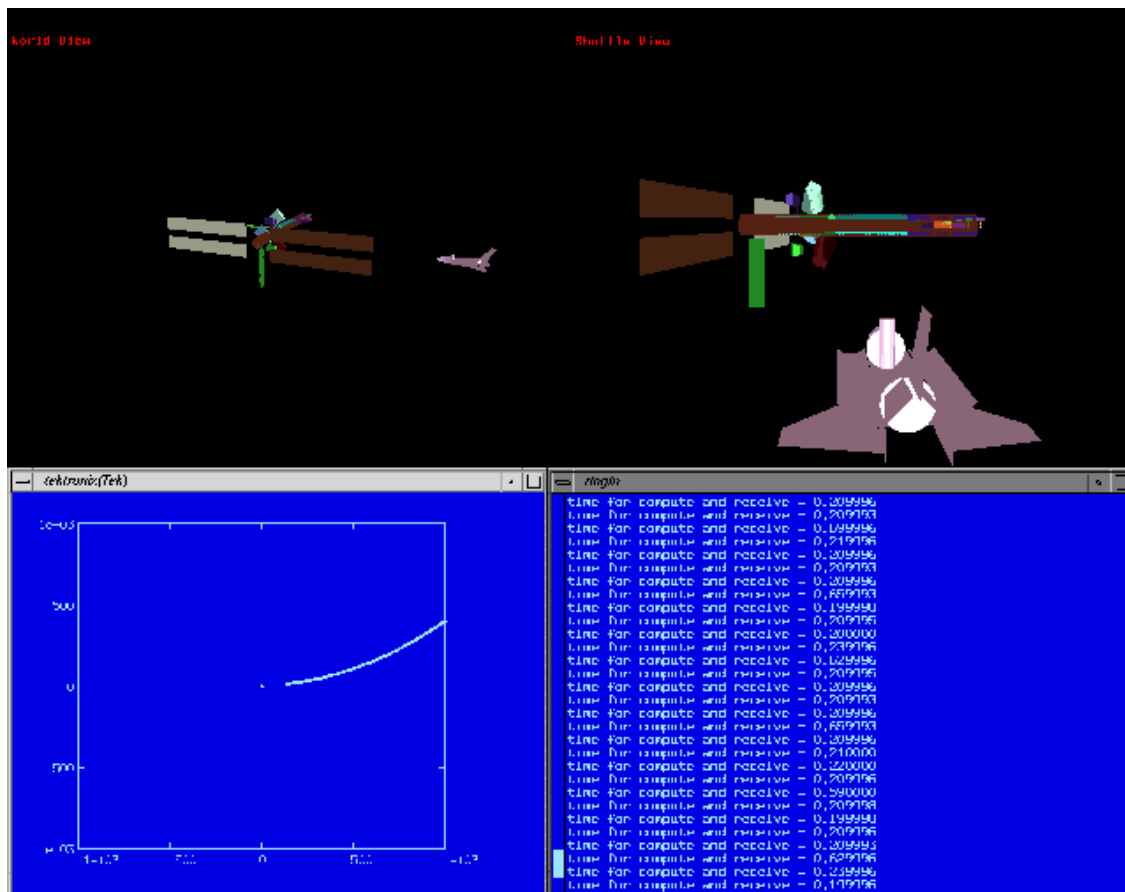


Figure 5: A screen image of the 3D graphical simulator. A movable view showing a close-up of the space station is in the upper left. The view from slightly above and behind an approaching shuttle is in the upper right. The plan view of the approach, perpendicular to the orbital plane, is in the lower left. The lower right is a numeric display of the state vectors and the communications channel.

iWarp parallel processor array can potentially provide massive computational power in a low-cost, low-power package. As a consequence, the iWarp offers significant potential for advanced space-based computing.

References

- 1) Reed, D., and Fujimoto, R., Multicomputer Networks, MIT Press, Cambridge, MA, 1987.
- 2) T. Fong, iWarp Evaluation Project, presented to the 1991 iWarp Forum, Arlington, VA.
- 3) Borkar, S., Cohn, R., Cox, G., Gleason, S., Gross, T., Kung, H.T., Lam, M., Moore, B., Peterson, C., Pieper, J., Rankin, L., Tseng, P.S., Sutton, J., Urbanski, J., and Webb, J., "iWarp: An Integrated Solution to High-Speed Parallel Computing," Proceedings of Supercomputing '88, 1988.
- 4) Grunwald, A.J. and Ellis, S.R., "Interactive Orbital Proximity Operations Planning System," NASA Technical Paper #2839, 1988.
- 5) Kaplan, M.H., Modern Spacecraft Dynamics and Control, John Wiley & Sons, 1976.
- 6) Taff, L.G., Celestial Mechanics, A Computational Guide for the Practitioner, John Wiley & Sons, 1985.
- 7) Borkar, S., Cohn, R., Cox, G., Gross, T., Kung, H.T., Lam, M., Levine, M., Moore, B., Moore, W., Peterson, C., Susman, J., Sutton, J., Urbanski, J., and Webb, J., "Supporting Systolic and Memory Communications in iWarp," Proceedings of the 17th Annual International Symposium on Computer Architecture, 1990.
- 8) Fong, T., and Soursa, R., "Real-Time Optical Flow Range Estimation on the iWarp," SPIE Applications of AI XI: Machine Vision and Robotics, 1993 (in press).
- 9) Wise, J.D. and Ciscon, L., "Telerobotic Interconnection Protocol", Rice University, 1990.
- 10) Sense8, "World ToolKit (TM) Reference Manual," 1991.

Acknowledgments

This work was sponsored by Tice DeYoung, DARPA CSTO, under ARPA#7718. The authors would also like to thank James Fincannon, of NASA's Lewis Research Center, for the Space Station Freedom and Space Shuttle models.