The Architecture of a Video Image Processor for the Space Station

S. Yalamanchili, D. Lee, K. Fritze, T. Carpenter, and K. Hoyme
Honeywell Systems and Research Center
Minneapolis, MN 55418

N. Murray
NASA Langley Research Center
Hampton, VA 23665-5225

Abstract

This paper describes the architecture of a video image processor for space station applications. The architecture was derived from a study of the requirements of algorithms that are necessary to produce the desired functionality of many of these applications. Architectural options were selected based on a simulation of the execution of these algorithms on various architectural organizations. A great deal of emphasis was placed on the ability of the system to evolve and grow over the lifetime of the space station. The result is a hierarchical parallel architecture that is characterized by high level language programmability, modularity, extensibility and can meet the required performance goals.

1 Introduction

A major goal in the design and deployment of the NASA space station is to enable crew members to effectively and efficiently use the resources of the space station. The number of anticipated scientific and commercial missions will place a heavy demand on these resources, one of which is crew time. Thus, facilities that enable crew members to perform their tasks efficiently, effectively, and safely are critical to the success of the space station. This paper describes the utility and feasibility of providing crew members with one such facility - a video image processor (VIP).

Initially, a crew member will directly control and be interactively involved with most activities, such as inspection, docking, experiment monitoring, and control. One of the problems with the man-in-the-loop scenario is that the human operator is frequently performing repetitive observations and control functions that do not exploit the unique capabilities of a human in space, namely, decision making, supervision, and creative thinking. Repetitive tasks are ideally suited for automation by machine. Such automation would free crew members for more demanding tasks as well make more efficient use of their time. An important technology central to automation and robotics is image processing. Video images may be processed to improve the quality for viewing purposes, provide cues in assisting an operator in some task, or provide some information to control other devices or alert the crew when necessary, as in automatic experiment monitoring.

Early on in the study [1], a surprisingly large number of applications were found that could benefit from the availability of an on-board VIP. The required algorithms and architectures necessary to support a VIP were found to be mature enough to make the concept of a VIP feasible. An examination of the functional requirements of image processing algorithms and the capabilities of current and future processors resulted in the conceptual design of a hierarchically structured, parallel architecture for a VIP. This paper reports the results of an effort to refine this conceptual view via simulation. The simulation studies were based on the requirements derived from an analysis of algorithms for space station applications. The design and validation of these algorithms are discussed in a companion paper [2].

2 Role of a VIP in the Space Station

The principal goal of a VIP is to increase the efficiency with which the space station resources are used. This would be achieved by automating certain tasks with the VIP as well as making some tasks more efficient. Although detailed requirements for various systems on the space station are still being developed, it is evident that a number of existing requirements support the concept of an on-board VIP. One requirement is that missions should be performed in a timely and safe manner. Missions include user experiments, user production activities, satellite servicing, and housekeeping tasks. A VIP may enable a crew member to perform more activities from a central location, such as a workstation. This would mean fewer extra vehicular activities (EVAs), which in turn makes the crew member more efficient and, in many instances, considerably safer. In addition, there are tasks that can be performed by a VIP that would result in faster execution (for example, providing automatic camera control), make the task easier (filtering of transmitted video containing noise), or eliminate the active participation of a crew member altogether (automatic experiment monitoring).

Another requirement relates to space station autonomy. Autonomy may be interpreted in two ways. The first interpretation is that an activity that can proceed autonomously from human interaction. A VIP helps in this case by performing functions relating to machine vision, freeing the user from constant interaction. The second is that the space station should operate as autonomous from ground support as feasible. In this instance, a VIP may be used in several ways. It can compress image data, allowing the relatively limited on-board storage capability to be used more efficiently. Thus, requests for data are more likely to be satisfied at the station without accessing ground-based archives. In this manner, a VIP can increase crew efficiency, making it less likely that ground-based personnel would be required to support the workload.

A related requirement is that the crew time necessary for housekeeping tasks should be minimized. As shown in the space station mission requirements report, one of the constraints on the number of active payloads will be the number of crew hours available to perform payload-related tasks. The assumption was made that with a crew of six, the equivalent of one person would be required just to perform housekeeping

chores. A VIP could help reduce this time by automating or supporting housekeeping activities, such as rendezvous and docking, space station inspection, and maintenance. During our examination of potential applications of a VIP [1], we generated the following set of tasks that could potentially utilise a VIP for increased safety, autonomy and efficiency.

- Construction
- Satellite servicing
- Rendezvous and Proximity Operations
- Docking
- Inspection
- Maintenance and Repair
- Payload Delivery and Retrieval
- Experiment Monitoring
- Data Management and Communications
- Training

A more detailed discussion of what role a VIP might play in each of these tasks can be found elswhere [1]. Finally a VIP is impacted by a need to evolve with the space station, primarily since it will not be possible to plan and accommodate all future processing needs. It also has to be compatible with the size, weight and power budgets that are constrained by the capabilities of the power generation subsystem and the payload capacity of the space transportation system.

3 VIP Algorithms

There were two goals for the algorithm selection process. First, the image processing techniques required must be mature and reliable, enabling a high degree of confidence in obtaining the desired functionality. This is particularly important due to the unique set of environmental, lighting and imaging constraints under which space imagery is acquired. Secondly, the algorithm suite should benefit a large number of applications. A cross reference between six major classes of image processing algorithms and the eight generic classes of space station applications is shown in figure 1.

These six families do not represent the entire breadth of the state of the art in image processing, but most of the image processing algorithms required for the automation of space station tasks belong to one of these families. In addition, algorithms in each of these categories are sufficiently mature for the design and build of a prototype system. This prototype system could be semiautonomous in that it could perform the majority of the data reduction necessary for a specific task and an operator would be required for verification/confirmation of the actions of the VIP. Typical applications in which a VIP may perform a task in semiautonomous mode are image enhancement/filtering, intelligent bandwidth reduction, and object velocity estimation for proximity operations. A more detailed discussion is presented in a companion paper [2].

4 VIP Architecture

As a result of the VIP I study, we recommended an architecture for the VIP, shown in Figure 2. It consists of a multiple-SIMD organization (level 1) followed by a multiprocessor organization (level 2). The multiprocessor will be designed to allow for the addition of processors specifically suited for symbolic processing, e.g., rule-based inference processing. The level-1 system may be viewed as a sequence of array processors. The first processor receives video data from the network interface. Each array processor stage may implement a specific image function, such as detector compensation, gray scale stretch, or digital filtering. The processed image may then be transferred to an image memory, which forms the input to another array processor implementing another image function, or may be transferred to the image memory of processors that perform the next higher level of processing (level 2 and level 3). They consist of more flexible multiprocessor systems that can be used to compute descriptions of areas of the image, e.g., regions of interest, boundary codes, statistics, etc. Processing at the higher levels may primarily deal with arrays of real or integer data (e.g., tracking and position estimation) or symbolic data (e.g., relational descriptions). These two types of processing are fundamentally different, but both are required at this level of processing. Our approach is to efficiently accommodate both modes of processing, loosely coupled through the use of a partitioned global address space. Architectures specifically suited for artificial intelligence execution are not considered at this time because of the immaturity of the concept. However, as research continues in this area and in artificial intelligence algorithms for image understanding, the addition of such processors may be allowed during the growth phase of the VIP.

The specific choice of a building block for each level must result in an overall organization that satisfies the constraints identified in the VIP I study. One set of constraints is due to the requirements of the Initial Operating Configuration (IOC). Depending upon the technology freeze date for IOC, current hardware, software, system and algorithm technology may not allow the development of a fully functional VIP within the anticipated size, weight and power constraints. The issue therefore is in phasing: being able to use that part of VIP that is useful and currently feasible, while provisions are made to allow it to evolve into a fully functional VIP. Certain critical portions (such as the level-1 architecture) may be included at IOC to perform those functions that are useful for the man-in-the-loop scenario. Then, during the growth of the space station, additional functionality could be provided using more advanced and stable technology, algorithms, and perhaps architectures. Phased implementation requires features of programmability, modularity, and field expansibility. The latter includes provisions for integrating

special-purpose devices into the architecture as their need becomes evident and their implementation becomes viable. Further, the implementation technology and architecture must be sufficiently mature to be considered for deployment. The VIP architecture proposed in this program is amenable to all of these constraints.

4.1 Level 1 Architecture

The VIP I study called for a synchronous parallel architecture that operated in single instruction multiple data stream (SIMD) mode and delivered in excess of 600 million operations per second (MOPS) performance. Furthermore, each processor would have access to neighboring processors' memories and would be microcoded. The rationale for these constraints can be found in the VIP I final report [1].

Our choice of the basic building block for the level-1 architecture is the Electro-Optical Signal Processor (EOSP) developed by Honeywell [3]. Its organization satisfies all of the above mentioned constraints. In addition, it possesses a number of other important features that make it a good choice for a VIP. The EOSP architecture was derived in a top-down manner from the requirements of real-time image processing algorithms. The result is a very high speed integrated circuit (VHSIC) chip set that can deliver up to 25 MOPS per processing element (PE) for low-level image processing tasks. Thirty-two PEs constitute a single stage of the EOSP, resulting in a computation rate of 800 MOPS per stage. It is optimized for image processing functions that are characterized by large volumes of data and repetitive arithmetic and logical operations over small neighborhoods of an image. Unlike many early image processing architectures, issues concerning the interface to different sensors and the implementation of image input/output (I/O) were also addressed early in the development. Thus, the EOSP is optimized to provide high throughput for raster scan imaging devices. Any algorithm that exhibits concurrency at the pixel level can be efficiently implemented on the EOSP.

The organization of the EOSP is illustrated in Figure 3. The architecture consists of a linear array of identical PEs, each with its own memory, controlled by a single common controller. This SIMD architecture minimizes the control overhead per PE, thus achieving extremely high computational rates within a very compact processor. In its current form, each PE has a 128-byte input buffer and a 128-byte output buffer. Local memory consists of 512 bytes accessed by a 16-bit arithmetic logic unit (ALU). Each of the I/O buffers is externally clocked. Thus, it is possible for data transfer into the input buffer, data transfer out of the output buffer, and processing of local memory contents all to be occurring simultaneously. This allows for a pipelined mode of operation in which images may be processed in real-time with storage requirements independent of image size. The EOSP architecture operates on an image on a line-by-line basis. Each image line is evenly distributed among the input buffers of the PEs and transferred to local memory. A sufficient number of consecutive image lines is stored to enable one line of image output to be computed. In the configuration of the above example, one line of a K x K window function can be computed by all the processors in parallel. Each processor computes M pixels of the output line. Next a new input line can be acquired, and one line of the computed image can be output. In this manner, one line worth of results is computed and output for every input image line from the sensor. This has the effect of "sliding" a K x K window over the input image. Data from the input buffer are transferred to the individual PE memories in parallel at the end of the scan line. Processed results in the PE memory, computed during the previous line input, are transferred simultaneously to the corresponding output buffers. Buffered results are read out synchronously with the input data entering during the next scan line. Input and output can be double-buffered for sensors that possess no dead time between lines (e.g., retrace time). This provides a great deal of flexibility in interfacing the EOSP to different types of sensors and architectures. Such a feature is especially attractive for the VIP since the functionality of the video network interface (VNI) (input to level-1 architecture) and the details of the level-2 architecture (output from the level-1 architecture) are subject to change. This feature is even more important if the VIP is to be deployed as the level-1 architecture only and is to subsequently evolve to include the level-2 architecture later in the life of the space station.

Sizing an EOSP system is done with respect to three features - processing throughput requirements, memory requirements and the I/O requirements. Whichever feature is the most demanding in terms of the number of processors required, dictates the size of the EOSP system. This essentially accounts for the fact that some applications may be throughput bound versus I/O bound or memory bound. Several examples are illustrated in Table 1. All pixel and neighborhood operations will be implemented in the level 1 architecture. This includes the color image enhancement algorithms [2]. Functioning brassboard versions of the EOSP PEs are available today. The technology and architecture can be considered to be mature by any future technology freeze date. Moreover, a great deal of familiarity with the EOSP systems has been obtained, establishing a degree of confidence in the ability to meet the projected performance goals. Experience has been gained and lessons learned in the design of the EOSP. For this and the reasons cited above, the EOSP architecture is an excellent choice as the building block for the VIP level-1 architecture.

4.2 Level 2 Architecture

Our earlier studies indicated that this level would require an 8 - 16 processor system delivering about 100- 200 MOPS with distributed task allocation, scheduling and synchronization. To understand the characteristics of the level-2 architecture, one needs to understand the algorithms that will be executed. The granularity of parallelism is relatively large (compared to those executed in the level-1 architecture), resulting in a number of concurrently executing tasks. The processing within a task is highly data dependent. As a result, interactions between tasks should be asynchronous. The volume of intertask communication is highly variable and can become the principal determinant of performance [3-4]. Thus, the first issue is the choice of interconnection topology. Once this has been chosen based on the requirements of the VIP algorithms, the architecture may be examined in greater detail to address issues of protocols, processor-specific features, and operating system features. The choices are limited only by one's imagination. However, we chose topologies that, in some sense, occur at extreme points in the spectrum of performance that interconnection networks can provide. At the same time, the choices were filtered by factors such as maturity, available experience with them, and how well we understood them. Our choice of families of topologies to investigate were multiple buses, hypercubes, and braided rings. These topologies are illustrated in figure 4

The next issue is one of analysis techniques. The level-1 architecture exploited fine grain parallelism in a synchronous mode of operation. Further, the algorithms are largely data independent. With such a fine understanding of the implementation of the computations, it is possible to analytically evaluate the architectural options. That is not the case with the level-2 architecture and algorithms. The high degree of variability

in the processing and communication requirements indicate that simulation is an appropriate means to determine the proper topologies. The Architecture Design and Assessment System (ADAS) tool set developed at Research Triangle Institute was used for this purpose. The tool set includes facilities for constructing models of communicating parallel tasks and parallel architectures. Further, tools are available to map communicating sequential tasks onto specific architectures and evaluate the performance of such a hardware/software system.

4.2.1 Simulation

The objectives for performing the simulation are multifold. First, we would like to verify that the proposed architecture design can meet the system throughput requirements, and that the specified image processing algorithms can be executed within the given time frame. Second, we want to compare the performance of several proposed architectures and topologies and analyze how they perform in executing the different algorithms. This would then provide guidance in selecting the appropriate architecture approach for the VIP design. Finally, we would like to use simulation as a tool to refine the architecture design. By varying the system size and characteristics, one can perform tradeoffs not only between interconnect topologies, but also in the number of processors and buses, processor speeds, and bus bandwidths. The ultimate objective is to enable us to select and derive a suitable architecture for the VIP design. The parameters we have chosen to study for the VIP simulation effort are categorized as follows.

- Network topology - Six interconnect network topologies were simulated: multiple buses with one two and three buses, hypercube, unidirectional and bidirectional braided rings.
- Communication bandwidth - Three separate bus speeds were used in the simulation: 2, 5, and 10 Mbytes/sec.
- Processor throughput - Three separate processor throughputs were used in the simulation: 2, 5, and 10 million instructions per second (MIPS).
- System size - System sizes of 4, 8 and 16 processors were considered.

In the description of the simulation, buses will be used to refer to both the multiple-access shared media, such as time shared buses, as well as point-to-point links, such as those used in the hypercube and ring organizations. The level 2 implements the components of the tracking and bandwidth reduction algorithm. Each of the computationally intensive components of this algorithm was studied in greater detail and parallel versions of these algorithms were derived and modelled with ADAS. These were,

- Monochrome Segmentation
- Boundary Tracing
- Linearity Filter
- Connected Components
- Silhouette Matching

For each of the above algorithms, conservative requirements on image resolution and other algorithm-specific parameters (e.g., size and number of objects) have been assumed in constructing the software graphs. Both of the above software and hardware systems are modelled in ADAS with directed graphs consisting of nodes interconnected by directed arcs. Nodes represent individual software operations or hardware functional elements, while arcs represent data flow between software operations or hardware components. The presence or absence of data or control is represented by tokens on the arcs. When an input condition is satisfied by the presence of specific patterns of tokens on the input arc a node "fires". It fires for some period of time after which tokens may be placed on some output arcs probably enabling another node. Once software and hardware graphs have been developed, the software graph is mapped onto the hardware graph to produce a constrained software graph. Since the software graph represents the algorithm executed by the hardware, the order in which the software graph nodes fire is determined by the structure of the underlying hardware graph. In particular, software nodes mapped onto the same hardware nodes can only be executed one at a time. Nodes represent the execution of a computation (transfer of data). The firing delays are therefore functions of the volume of computation (data) and the processor speed (link or bus bandwidth). The simulation sequence considers the range of values for processor speeds (link or bus bandwidths). Some examples of hardware and software graphs are shown in figure 5. The simulation sequence proceeds as follows.

1. Construct software and hardware graphs. The software graphs represent the image processing algorithms to be executed, while the hardware graphs represent the architectures and constraints of the hardware system.
2. Place appropriate weights on software nodes. These weights include the various assumed characteristics, such as delays, amount of processing requirements, processor throughputs, and network link bandwidths.
3. Constrain the software graph execution by mapping the software graph to the hardware graph. This involves assigning various software modules (algorithms) to the different hardware modules (nodes).
4. Execute the constrained software graph and collect execution statistics.
5. Modify the weights in step 2 to effect a change in the parameters of interest and repeat the sequence.

For the purpose of evaluating the results, the following performance measures were generated by the simulation.

- Latency - This is the time for one execution of the complete software graph (algorithm).
- Average processor utilization - This is the average percent of execution time processors are busy.

- Maximum processor utilization - This measure is the maximum percent of execution time that a particular processor is busy. It identifies the presence of bottlenecks.
- Variance of processor utilization - This provides a measure of balance in processor utilization and thus, the distribution of the computation load.
- Average bus utilization - This is the average percent of execution time the buses are being used.
- Maximum bus utilization - This identifies the presence of communication bottlenecks.
- Variance of bus utilization - This measure indicates the distribution of the communication load.

To control the ADAS simulation sequence and facilitate the generation of the performance measure statistics, a simulation manager was developed. The simulation manager is the core of the simulation management facility. It essentially controls the iterative execution of the simulation. The details of the simulation management facility can be found in [6].

4.2.2 Simulation Analysis

To facilitate the analysis of the simulation data in selecting a suitable VIP architecture, we decided to evaluate the performance of the various designs based on the following performance metrics.

- Low latency - Latency is the major criteria in evaluating the performance of a design. The system throughput must be above some minimum threshold in order to satisfy the basic timing and processing requirements. Beyond that threshold, low latency may be traded off against other considerations.
- Balanced processor utilization - The preference here is to evenly distribute the processing load among the processors as much as possible, thereby avoiding the presence of bottlenecks and reducing the severity of single-point failures. This can also serve as an indication of how growth and fault tolerance can easily be achieved with the design.
- Balanced bus utilization - The preference here is to avoid communication bottlenecks and severity of single-point failures. Again, this can serve as an indication of the ease with which fault tolerance and future growth may be accommodated.
- Latency and utilization improvement - This is the differential of the latency or utilization as a function of some architectural parameter. This measure is used to identify points of diminishing returns. For example, a doubling of processor speed may produce only a 2% decrease in latency. In that case, the cost of designing a faster processor may not be worth the added speedup. A similar argument can be made for utilization and, in fact, for most parameters. Another view is that this measure indicates the sensitivity of the latency and utilization metrics to various architectural parameters.

In addition to the above performance metrics, we also made the following empirical assumptions concerning the VIP design requirements.

- The design shall provide a processing throughput margin of 100%.
- The design shall provide a communication bandwidth margin of 100%. These first two assumptions allow for growth in algorithmic requirements and other unexpected overheads.
- The design shall allow the presence of spare processors and spare buses. This enables the design to provide for fault tolerance as well as growth capabilities.
- The VIP design shall execute the tracking and bandwidth reduction algorithm at the rate of about one image per second. This assumption is more of a desire than a requirement. In reality, considering the anticipated applications of VIP in the space station, a processing rate of one image per every few seconds may even be acceptable for most applications.

With the above initial assumptions and performance metrics in mind, the simulation data were analyzed and evaluated. A software graph for the tracking and bandwidth reduction algorithm was constructed, and its execution was simulated on the various architectural organizations. This includes parallelized versions of the selected components. The size of the search space of the architectural alternatives is fairly large. There are six organizations - three for the bus-based systems, one for the hypercubes, one for the unidirectional rings, and one for the bidirectional rings. For each organization, there are three system sizes (4, 8, and 16 PEs), three processor speeds (2, 5, and 10 MIPS), and three bus bandwidths (2, 5, and 10 Mbytes/sec). Thus, there are (6x3x3x3) or 162 distinct possible architectural solutions in this formulation. For each possible architectural solution, the parameters of interest are measured and tabulated. These results were examined manually to apply the chosen metrics and select acceptable solutions. The result reveals that a configuration with 16 processors, each with a processing speed of 10 MIPS and a dual-bus network with bus speeds of 5 Mbytes/sec, comes closest to meeting all of the empirical assumptions and performance metrics mentioned above. The simulation performance data for this architecture are summarized in Table 2.

The simulation data also indicate that the hypercube configuration (N = 4), with 16 processors at 10 MIPS each and bus speeds of 5 Mbytes/sec, is also a viable alternative. The final latency value for configurations with the hypercube design is shown in Table 3. Currently, the bus-based approach is preferable to the hypercube approach mainly because it is a relatively more mature and well-understood architecture. In this respect, the bus-based approach represents a low-risk approach. While the hypercube technology has now become a commercially viable product, improvements are rapid and continuous. The network is inherently fault tolerant through the presence of mutiple paths between nodes, but it is not immediately obvious how that feature may be efficiently exploited. The area that needs the most attention is operating system support. Efficient internode communication and global resource allocation strategies are lacking and are the focus of several research efforts by both commercial and academic organizations. By comparison, software in general, and operating systems in particular, are much more mature in bus-based systems. Further, increases in performance by the addition of one, two, or a small number of modules are straightforward in bus-based systems. Generally, the number of modules is doubled to maintain the connectivity of the hypercube. The addition of a smaller

number of processors is not straightforward. Thus, while simulation experiments indicate that the hypercube is an acceptable solution, practical considerations indicate that bus-based simulations are preferable. Hence, the 16-processor, two-bus system is the choice for a VIP.

4.3 System Issues

System issues can now be addressed in more detail with respect to this specific organization. System issues relate to three aspects of the VIP. The first is the interaction of the VIP with its environment. This is defined by the functionality of the VNI. The second concerns the software requirements, and the third, the hardware requirements.

4.3.1 Interaction with Environment

The VIP is intended to support bidirectional transfer of video data to and from devices on the space station. The VIP processes raw video data from a variety of video sources - including video cameras, video storage devices, and uplink video - and transfers processed, filtered, and enhanced images to various sinking devices on the space station. In order to specify the functionality of the VNI, it is necessary to make some assumptions about the operating environment. For example, what is the nature and frequency of traffic to and from the VIP? It is clearly infeasible to consider all possibilities. Therefore, we focus on what we feel will be the most prevalent scenario for the use of a VIP: a crew member controlling and using the VIP from a multipurpose applications console (MPAC). For example, cameras possibly mounted outside the space station may transmit images to the MPAC. These images may be redirected from the MPAC to the VIP for enhancement for viewing purposes. Alternatively, the VIP could receive images directly from cameras (under MPAC control) and relay results to the MPAC on detection of a specific event, e.g., in automatic experiment monitoring. In such a scenario, the functionality of the VNI would be determined by the nature of the interaction with the MPAC and by the operation and type of communications media between the MPAC and the VIP.

The MPAC will be one of the primary interactive display devices on the space station. Images will be displayed in the video/graphic/text application display area of the MPAC, and the console will present a mixture of information types, such as graphic, tabular, textual, video, discrete, etc. The advanced Work Package 2 implementation guidelines [7] demonstrate a preference for the display of color image data. However, the capability must exist for handling both color and monochrome video data formats. From the point of view of the interaction with a VIP, it is assumed that the MPAC will provide for the buffering of processed images since most functions are not processed at the image data rate of the MPAC, and graphics and image database functions will not be provided by the VIP.

The space station data management system (DMS) can support the requirements for bidirectional communication between the VIP and MPACs. Data transfers between the VIP and the MPAC involve the transfer of commands and images from the MPAC to the VIP and status from the VIP to the MPAC. Commands take the form of enable/disable for the VIP, diagnostic commands, as well as a selection of algorithms. The volume of communications for such a transfer is expected to be low, 200 to 300 bytes every 1/15th second. Thus, tolerable network latencies are determined by the interactive nature of the processing. It is the image transfers that place demands on the bandwidth of the DMS. These are high in volume and place stringent demands on whatever communication network is available. Two options may be considered in determining how this traffic may best be handled. The first is to use all digital transmission and the space station DMS. The second is to use a separate analog network and retain the images in analog video form. Both options are viable and possess advantages and disadvantages. However, it should be noted that the choice of one or the other does not impact the functionality or operation of the VIP, but only affects the VNI.

4.3.2 Hardware Issues for VIP

Several distinct hardware issues arise in the organization of the VIP. These are related to the four principal components of the architecture: the VNI, the level-1 PEs, the interface between the level-1 and level-2 architectures, and the level-2 PEs. The functionality of the VNI and issues related to it have been discussed in the previous subsection. The EOSP architecture is an existing system, and most, if not all, hardware issues relevant to the VIP have been resolved. The operation of, and interface requirements to, an EOSP architecture are defined [3]. Issues related to the remaining two components are discussed in this subsection.

This interface is physically a bus that can accommodate data transfers at least at the sensor rate. Operation of this bus is embedded in the functionality of the VNI and the bus interface units of level-2 PEs. This bus, in addition to serving as the physical interface between the level-1 and level-2 architectures, also is the interface between the EOSP and the VNI for output of image data to the network. This bus is interfaced to the output buffers of the EOSP PEs. Since these buffers are externally clocked, some degree of freedom is available in designing the bus to interconnect the EOSP stages, the PEs at the next level, and the VNI. This sensor rate bus provides a parallel, multidrop data and message transfer medium and is a custom bus defined to meet the requirements of the VIP. The data transfer bus is a 16-bit parallel bus and thus is matched to the word width of the EOSP I/O data paths. The 16-bit bus also provides sufficient bandwidth for the anticipated data transfers. The control bus portion must provide signal lines for interrupts, bus arbitration, and broadcast. Given the block structured nature of data transfers, multicycle arbitration schemes with timeouts are probably preferable since the control overhead will be amortized over the size of the data transfers. In addition, with proper design of the flow of control, it is unlikely that all three components would be simultaneously requesting the bus. The interrupt facility would also be used to synchronize the transfers between the EOSP and the level-2 architecture. Use of a command facility for the sensor rate bus could eliminate the need for an address bus at the level-1 interface. The majority of data transfers across the sensor rate bus are block oriented rather than byte or word oriented. The EOSP output data is transferred across the sensor rate bus on a horizontal scan basis. Data transferred from the EOSP to the VNI is also based on the scan line as the unit of data transfer. A command code may be active during the beginning of a block transfer or for the duration of a transfer depending on the command type, e.g., beginning of a scan, EOSP microcode start address, end of scan, etc.

The two principal components of the level 2 architecture are the PEs and the multibus system interconnecting them. The architecture and its interface to the EOSP are illustrated in Figure 6. Each PE consists of a processor module with local memory, a global memory module, and bus interface units. The processor (with local memory) interfaces to the sensor rate bus and accesses the two inter-PE buses through the associated global memory element. Such an organization has several advantages. From the point of view of developing a testbed, all of the components

and interfaces can be implemented with available standardized commercial components. This could actually continue to be the case, with some modifications, for a deployed version of the VIP. From a performance viewpoint, interspersing the processor between the global memory element and the sensor rate bus is crucial. This global memory element can provide performance equivalent to a locally accessible private memory for the local processor. At the same time, this element is available as globally accessible shared memory via the dual buses, and thus functions as a true shared memory since the local processor is not in the path for global memory accesses from remote PEs. The price paid for this generality is that the processor interface unit is in the path for data transfers from the EOSP, and the processor and memory share interfaces to the two inter-PE buses. Considering the synchronous, predictable, block structured nature of communication between the level-1 architecture and the PEs, this is not considered a significant disadvantage. Each memory element consists of fast access, static, random access memories (RAMs). Single-port access to the memory is provided by the local bus interface and the two level-2 global bus interfaces. All three bus interfaces would contend for port access on an equal priority basis.

Each PE consists of a processor, bus interface units, local bus systems, and a global memory element, as illustrated in Figure 6. The processor consists of a generic, 32-bit, single-chip computing element, such as the Motorola MC68030. Elements such as this can provide the computing power necessary to satisfy the throughput requirements determined by the VIP ADAS simulations. A complete set of software development tools, such as compilers, assemblers, and debuggers, is also typically available for such elements. The availability of such mature hardware/software environments is particularly advantageous for the testbed development phase.

The processor bus interface unit controls data transfers between the sensor rate bus, the PE and local memory, and the global memory element. Data may be transferred directly from the sensor rate bus to the global memory element. Data transfer may also occur between the local program memory and the global memory element. Each hardware node interfaces with a number of bus structures. The first is the sensor rate bus interface. The second is the intraprocessor bus system between the processor, local memory, and the bus interface unit. This would likely be a generic asynchronous bus interface well suited to interconnection between the generic processor and local memory. Finally, there is the local bus system between the processor unit and global memory element. A standard, 32-bit, asynchronous bus architecture, such as the VME bus [8], would suffice for this latter bus. An asynchronous bus structure for this local bus simplifies the bus protocol and allows for fast arbitration and capture of the system bus. This feature lowers PE dead time during a bus arbitration phase for single-word and short block transfers. Use of block transfers after the bus arbitration phase supports block-level direct memory access between the sensor rate bus and the global memory element.

A two-bus system architecture has been suggested for the VIP level-2 architecture. Global memory interconnection to the level-2 buses 1 and 2 is depicted in figure 6. There are a number of important qualities that the level-2 bus should possess. From the simulation studies, this bus system should provide a minimum average data transfer bandwidth of 5 Mbytes/sec. This performance figure is not difficult to achieve with many standardized bus architectures. A 32-bit data transfer bus width is preferred. This prevents packing and unpacking of 32-bit data that will be typically required. Further, the bus architecture should be processor independent and should allow a fairly large number of modules to interconnect to the buses. The current system calls for 16 PEs. In addition to the processor hardware nodes, there may be communications controllers that connect the VIP to the space station DMS via the level-2 bus.

A standard bus architecture that could meet the requrements for a VIP level-2 bus architecture is the Multibus II [9] bus structure. The Multibus II system bus is a high-performance, 32-bit bus capable of supporting up to 20 independent modules. This bus system is synchronous, supports processor independence, and supports block-level data transfers.

4.3.3 Software Issues

Currently under development is a microcode compiler for the EOSP and Distributed ADA [10], a candidate for the level 2 architecture. Capabilities have been successfully demonstrated on restricted problem sets. When finished they will enable the full VIP (level 1 and 2) to be programmed in ADA. With respect to operating system issues, we feel that a modification of an existing operating system such as Hunter and Ready's VRTX system will provide the functionality required of VIP. Schemes for distributed task allocation and scheduling are either handled within distributed ADA or have been developed [6]. Finally existing schemes are applicable for handling cache coherence and other problems that may arise. This is primarily due to the embedded nature of the VIP applications.

5 Concluding Remarks

Overall, a VIP will serve as a valuable utility to crew members on the space station, enabling them to efficiently accomplish their mission objectives and improve use of the space station resources, especially crew time. The architecture of VIP is based on relatively mature technology, one that will be stable before any future technology freeze date. Many of the systems issues can be resolved with existing hardware and software technology. The overall effect is one of comparatively low risk with the prospect of increased efficiency in many space station applications.

6 References

1. Honeywell Systems and Research Center, *Space Station Video Image Processor Concept Development*, Final Report, 1985.
2. P. Symosek et.al., *Knowledge-Based Vision for Space Station Object Motion Detection, Recognition, and Tracking*, Proceedings of the NASA Workshop on Space Telerobotics, Pasadena CA., January 1987.
3. Honeywell Systems and Research Center, *Electro-Optical Signal Processor User Manual*, 1986.
4. B. Lint and T. K. Agerwala, *Communcation Issues in the Design and Analysis of Parallel Algorithms*, IEEE Transactions on Software Engineering, vol. SE-7, March 1981, pp.174-188.

5. S. H. Bokhari, *On the Mapping Problem*, IEEE Transactions on Computers, vol. C-30, March 1981, pp. 207-214.

6. Honeywell Systems and Research Center, *Video Image Processor for the Space Station*, Final Report, November 1986.

7. *Space Station Work Package 2 - Definition and Preliminary Design Plans*, Habitability/Man Systems Report, vol. 13, NAS9 - 17365 DR - 02, 1986.

8. Signetics, VME Bus Manufacturers Group, *VME Bus Specification Manual*, rev. B, August 1982.

9. Intel, *Multibus II Bus Architecture Specification Handbook*, 1984.

10. Honeywell Systems and Research Center, *Honeywell Distributed ADA Project*, Status Report 1985.

	Color Image Enhancement	Tracking	Surveillance	Identification	Proximity Operations	Bandwidth Reduction
Construction	X	X		X	X	
Satellite Servicing	X	X		X	X	
Redezvous and Proximity Operations	X	X	X	X	X	
Inspection	X	X			X	X
Payload Delivery/Retrieval	X	X	X	X	X	X
Experiment Monitoring	X	X		X		X
Data Management and Communications	X	X		X		X
Training	X	X		X		X

Figure 1. Cross reference between applications and algorithms

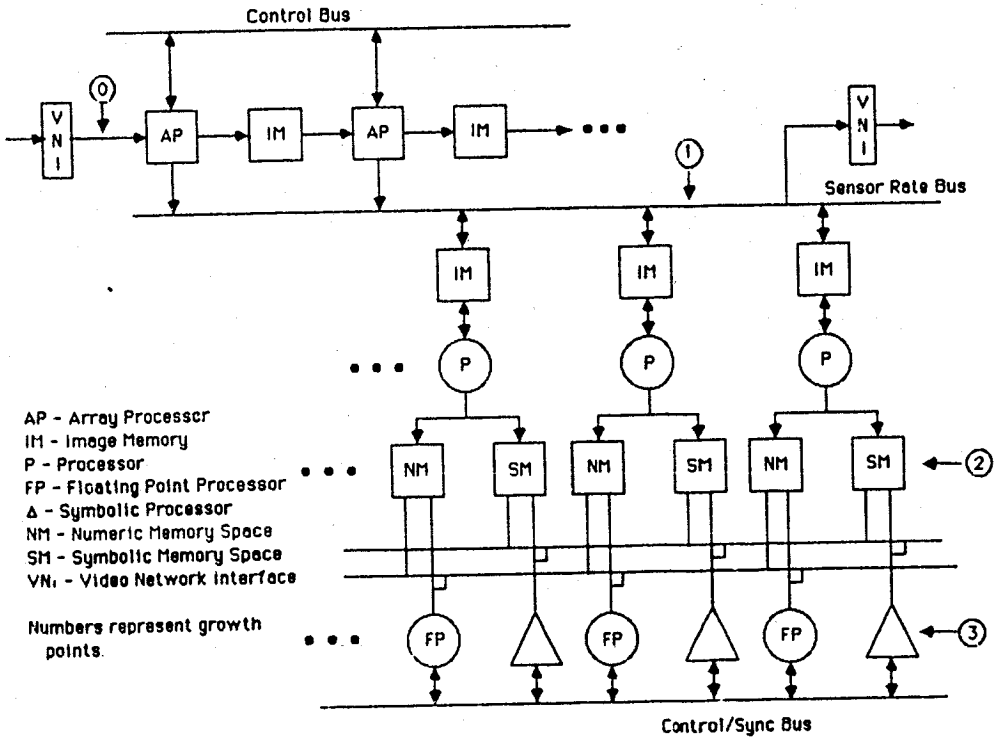

Figure 2. VIP Conceptual Architecture

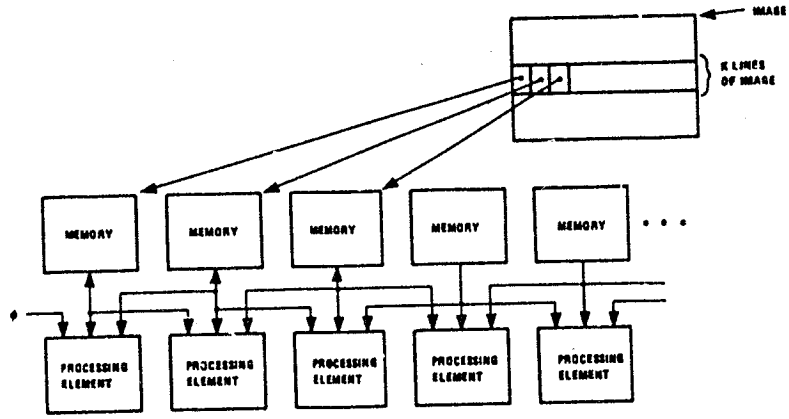

Figure 3. Organization of the EOSP

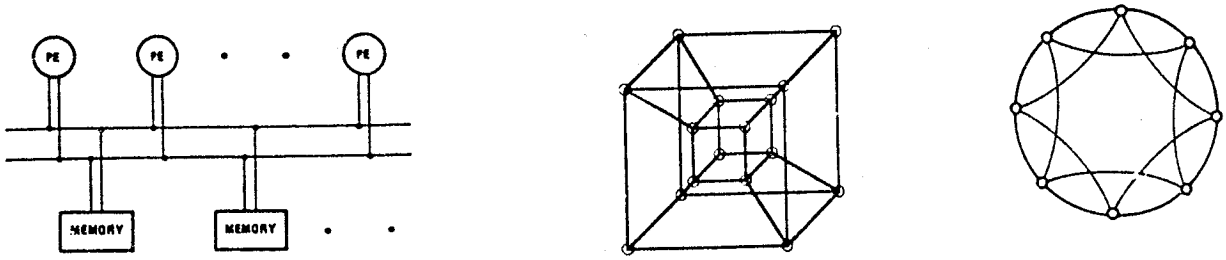

Figure 4. Bus oriented, hypercube and braided ring organizations

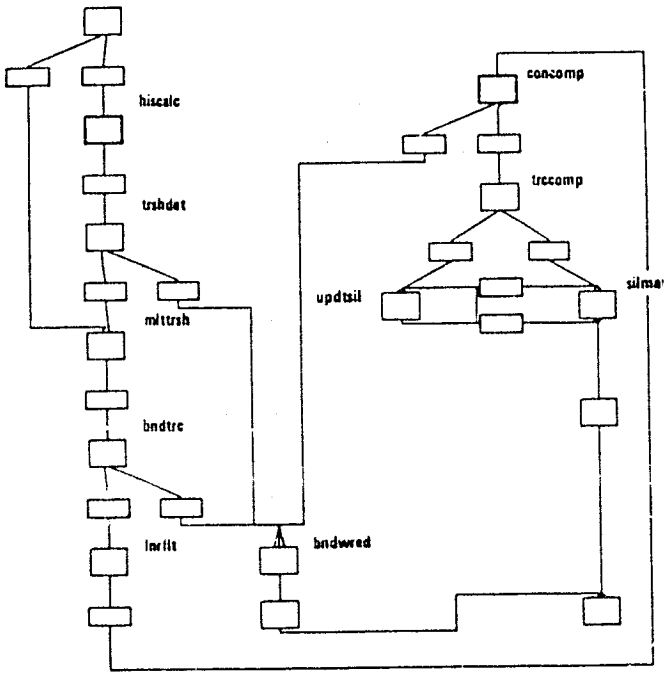

Figure 5. ADAS graph of the tracking and bandwidth reduction algorithm before parallelizing the components

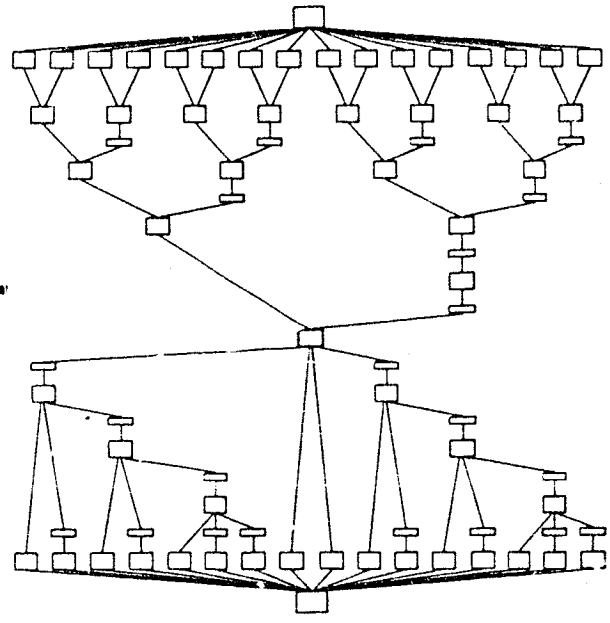

Figure 5 cont. Example ADAS software graph corresponding to a parallelized component