Progression of an Open Architecture: from Orion to Altair and LSS

Mitch Fletcher
Chief Systems Engineer
Human Space Business Segment

Honeywell, International
Defense & Space Electronics Systems - Glendale
19019 North 59th Avenue, Glendale AZ 85308
PO Box 52199, Phoenix AZ 85072-12199
Phone: 602-561-3158
FAX: 602-561-3076
E-mail: mitch.fletcher@honeywell.com

Progression of an Open Architecture: from Orion to Altair and LSS

Abstract

NASA has embarked on a very ambitious plan for space exploration over the next several decades. The cornerstone of this activity is the Constellation Program. Even with the retirement of the Space Shuttle and the NASA development budget growing from approximately $3.5 Billion to approximately $7 billion in 2011, this budget will require a different model for NASA implementation than was used on the Space Shuttle or the Space Station. The Constellation "system of systems" that contains seven major elements must be implemented within approximately twice the budget that a single space station element was implemented. Over the past decade, industry managers including NASA managers, have come to accept as axiomatic that the use of open systems reduces cost, decreases schedule, and eliminates risks to a program. However, the term, "open systems architecture" invokes a variety of interpretations. To some it implies no proprietary components. To others it implies adherence to documented standards. Still others see it as implying plug-and-play features. Honeywell has worked with NASA and other customers over the last five years to understand the voice of the customer relating to the benefits of varying level of open architecture. As Honeywell has developed the detailed architecture for the Orion vehicle, the team has strived to create an open architecture approach portable to Altair and beyond. This paper/presentation details the open features of the 6th Generation architecture and the ongoing enhancements to the Orion in work to finalize an open system to proposed throughout the Constellation architecture. In addition, the path to a real time computational platform useful throughout the Fault-Tolerant Spaceborne Computing community is extrapolated and discussed. This paper concludes with observed interpretations of the meaning of "open systems architecture" and the benefits to NASA within the long-life Constellation Program

Constellation Background

Over the decade there has been an evolving vision for the future of the National Aeronautics and Space Administration (NASA) human space objectives. The first step in this evolution was the Space Launch Initiative (SLI). According to Art Stephenson, director of NASA's Marshall Space Flight Center, Huntsville, Ala., "The Space Launch Initiative (was) a comprehensive R&D effort that provides technology developments that dramatically increase the safety, reliability and affordability of space transportation systems. The strategic goals of SLI were to develop concepts and the technologies to allow creation of a next-generation Reusable Launch Vehicle (RLV). This RLV would be designed such to reduce the risk of loss of crew to approximately 1 in 10,000 missions and to lower the cost of delivering payloads to low-Earth orbit to less than $1,000 per pound. In early 2003, the NASA refined their vision and funding to better support the SLI vision as shown in Figure 1. As part of this change, the broad SLI was split to focus on both short term and long term solutions. The short-term solution became know as the Orbital Space Plane (OSP) concept. The OSP was to focus on a system capable of providing crew rescue from the International Space Station (ISS) as early as 2008 and crew transfer in the 2010 time frame.

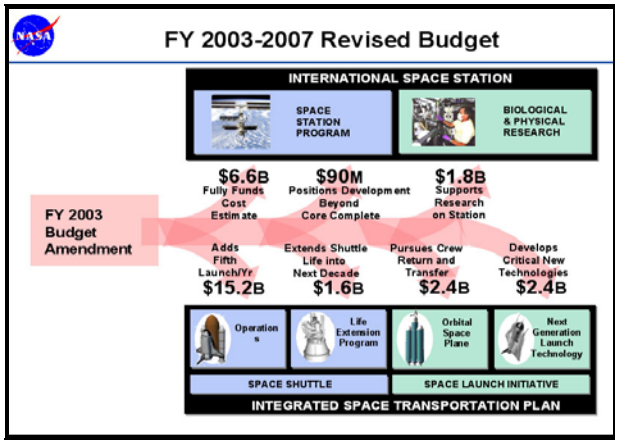

Figure 1 – *NASA 2003 Five Year Budget Plan*

The next evolution in the NASA manned space policy came on January 14, 2004 when President Bush announced the Exploration Systems vision as shown in Figure 2. This policy presented three goals. The first goal was to complete the International Space Station by 2010 and focus our

future research aboard the station on the long-term effects of space travel on human biology. The second goal was to develop and test a new spacecraft, the Crew Exploration Vehicle, by 2008, and to conduct the first manned mission no later than 2014. The final goal was to return to the moon by 2020, as the launching point for missions beyond. Included within these goals are the retirement of the Space Shuttle Fleet at the end of ISS complete and a series of robotic missions to the lunar surface, starting no later than 2008, to research and prepare for future human exploration. The ultimate goal of the new vision is to embark on human missions to Mars and to worlds beyond.

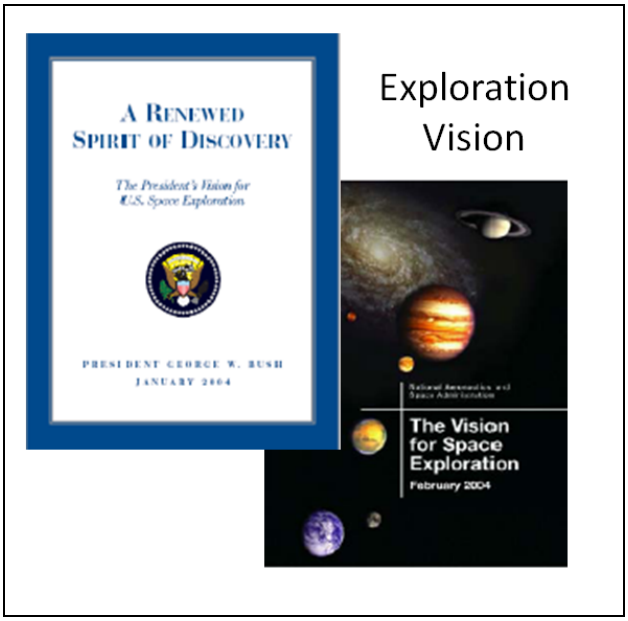

Figure 2 - *Current NASA Space Exploration Vision*

Even now the mission goals and objectives are changing. President Obama recently called for a review of the Constellation mission. This review, as reported by multiple press reports was prompted as an examination mission goals and the ability to complete these missions within the projected cost and schedule. The recent 2010 NASA budget has included substantial increases for both the Ares 1 and the Orion projects, while postponing the financial decisions of the Altair and Lunar Surface Systems programs to the result of the Augustine Commision report.

Throughout the evolution of the NASA space exploration vision, one requirement has remained constant; the need for a flexible and re-configurable computing infrastructure to host, control, and manage the systems that will enable the envisioned missions. Throughout the evolution from SLI to project Constellation, the need for a computing platform has evolved from simple vehicle level avionics control to a system of systems control architecture for both transportation and habitats. Starting in 2000, Honeywell recognized that a simple flight computer (a then current research project) was not addressing the issues associated with advanced human rated systems. In 2001, Honeywell took the first steps to develop the first advanced avionics system to address suitability of various approaches to meet the needs of an advanced human rated avionics system.

The NASA goal for avionics systems was to have an affordable system, both acquisition and life cycle that was at least as reliable as the current shuttle. In addition to simple vehicle control, the avionics platform must now host autonomous operation, enable system level operational reconfiguration, and support a higher level of Integrated System Health Management (ISHM). Due to longer mission times, all of these functions must be hosted with an ever-increasing demand for a composite of integrity and availability. To meet the goals of the Constellation Program, the cost of the avionics platform, including hardware and software must be drastically reduced from the costs incurred by the Space Shuttle and ISS programs.

The NASA vision for reducing the ultimate cost of the Constellation program was to rely heavily on Commercial off the Shelf (COTS) based systems to reduce the design costs of the advanced avionics systems. The cost benefit of these systems was seen to exist in their open architecture, where multiple suppliers will compete with each other to keep the cost down. There are several issues with this approach: COST designs evolve every 18 months, requiring a large logistic effort throughout the life cycle, they require part replacement to meet the NASA radiation environment, and the nature of the design implementation eliminates the fault isolation zones in all previous spacecraft designs, thus making these COTS based systems far less reliable than the requirement of being as reliable as the Space Shuttle.

Open System Definition

As noted by NASA in their desire for an open system, intuitively there are several apparent advantages to an open system architecture. Generally these advantages respond to problems that have plagued systems in the past. Apparent benefits include solutions to problems associated with single suppliers, NRE costs, maintenance costs, and upgrade costs.

Single Supplier. Open systems avoid the constraint of relying on a single supplier. Several risks are associated with a single supplier including the possibility of the supplier going out of business, the supplier increasing price due to their monopolistic position, and the supplier discontinuing support for older versions of a product.

NRE Costs. Open systems appear to reduce development costs. Using COTS components eliminates the need for new development. Open systems rely on the likelihood of multiple suppliers, fostering competition that leads to lower prices. Integration of COTS components are often handled by the suppliers, producing a list of compatible products for use on the project.

Maintenance Costs. Open systems increase the prospect that there is a pool of experienced users, decreasing the need for training efforts. Secondary support products such as development and maintenance tools are likely available as well. Support organizations, including supplier technical support, is often available for a yearly licensing fee.

Upgrade Costs. Many of the advantages associated with development and maintenance costs associated with open systems apply to upgrade costs as well. Competition drives product enhancements by suppliers at low or no cost to the project.

While all of these benefits seem intuitive, experience has shown that reality does not always match theory. When decisions are made on open system principles, it is important that system architects select those practices that best meet their specific goals, while accounting for potential associated problems.

Dimensions of Openness

In system architectures, the definition of "open" is hazy at best. Some are more generally accepted than others, but none are universally acknowledged. Among the more common definitions are:

- Documented Standards
- Widely Used Standards
- Non-Proprietary Interfaces
- Plug and Play
- Commercially Available End Items
- Commercially Available Development Tools
- Long Life Availability
- Open Source Code for Software and Firmware

In an attempt to reach the compromise between the desired COTS solution for an advanced avionics system and a solution that meets the actual requirements of affordability, availability, reliability, maintainability, and functionality while using available technology, Honeywell completed extensive research into aspects of "openness" that would result in meeting NASA's needs and desires. For each category of open systems, there are both benefits and potential issues. There is no universal approach to open architectures that is guaranteed to reduce cost or schedule. Each requirement driving toward increased openness should be carefully analyzed and a cost-benefit analysis performed. Only those open architectural principles that will truly result in decreased cost or schedule should be imposed on a project.

The team of Honeywell senior systems engineers performing the analysis of open system architectural principles concluded that the following categories are most likely to produce significant savings and should be seriously considered when designing a new system:

- Widely Used Standards
- Non-Proprietary Interfaces
- Commercially Available

The details of this study are documented in *"Open Systems Architecture - Both Boon and Bane"*[1] (IEEE/AIAA 10.1109/DASC.2006.313746). While other categories may also produce significant savings on a case by case basis, generally they are less likely to be cost effective. In all cases a serious

analysis should be performed to determine the optimal level of openness for each project. The bottom line is, open systems do not always provide cost or schedule relief, and in many cases cause increases over the life of the program.

The Honeywell goal over the past decade has been to create an avionics system approach that used the best of "open architecture" and proven advanced approaches that result in the cost savings that NASA needs while not creating a proprietary approach that locks NASA into a single provider. Honeywell believes that a valued NASA supplier can provide value in continuous execution of open system solutions, provided that those solutions truly meet the NASA goals and objectives. Honeywell believes this because this is the commercial model for success that has been the mainstay of Honeywell business for the last forty years.

IMA Definition

At the simplest abstraction, the requirements for a future NASA advanced avionics system involve receiving sensor input, processing that data, and actuating effectors. Ideally, this same unit would be the host for any autonomous activity, health management, and housekeeping functions. To reduce life cycle costs, this "flat" architecture would employ commercial interfaces to all input and output and have the hardware completely de-coupled from the hosted software.

To meet low initial acquisition costs demanded by the current NASA space exploration 5-year budget, and to meet the desire of low life cycle cost challenges, it is essential to rely heavily of existing commercial standards for hardware and software. In fact, it would be ideal to use existing Commercial Off-The-Shelf hardware if feasible. Unfortunately, as previously described herein, attempts to create high reliability systems using COTS equipment have failed to be successful. To meet the desired NASA cost required to implement all the Constellation requirements (including technical, cost, and schedule), an advances avionics system are likely to include the following:

1. *A system that is expandable and/or re-configurable in the future*
2. *A system that uses open architecture concepts to allow flexibility within the implementation*
3. *An architecture that allows third party participation either in the development of the avionics hardware or at a future time*
4. *A low cost system throughout the lifecycle - this implies low development cost, low integration cost, and future low cost of ownership*
5. *Time & Space Partitioning and Fault Tolerant Middleware*

Honeywell has executed a multiyear project and adapt the best features of existing commercial systems to create a cost effective advanced avionics system architecture that looks, feels, and implements like the flat architecture with features that allow implementation using low-cost microprocessor-based units providing the flexibility of a distributed data system. This implementation is an adaptation of the DARPA Fifth Generation System they branded Integrated Modular Avionics (from a presentation by Ron Szkody on 29 May 1996 to the Integrated Sensor System (ISS) Open System Architecture (OSA) Joint Task Force (sponsored by United States Air Force Wright Laboratory /AAST-30). This system includes more than just the hardware. The requirements also include the software components, tools, and processes necessary to effectively develop, deploy and maintain the system.

This approach will meet the highest levels of integrity and availability at the lowest initial acquisition cost along with minimal cost of ownership and upgrade.

A Backbone Designed for both Flexibility and Availability

The key to ultimate flexibility and re-configurability is to architect a system where all information, whether it be I/O data or computationally derived data, is available to every aspect of the data system. The simplest method to achieve this goal is to implement a centralized system where all the data is in core memory available to the one and only processing unit. This implementation meets the simplistic requirements for a computational previously described herein. The drawback is that this is a circa 1960 implementation which does not support the fault tolerance, scalability, and availability requirements of a modern system.

The alternative to this is to implement a system approach wherein distributed components are coupled together in an architecture that provides the advantages of a centralized system. The key architectural element is the ability to make all the system data available to all of the distributed processing elements within the system through a carefully orchestrated sequence using a high integrity Backplane for single box systems, or through a Virtual Backplane™ for a multiple box system. The simplistic realization of the Integrated System data architecture is shown in Figure 3. As illustrated, the centralized computing engine is broken up into an arbitrary number of distributed computing engines that may or may not contain an interface to I/O. I/O and ancillary functions such as communication and data storage can be made available throughout the system with or without the computational element. Any of the compute elements can be assigned to execute any portion of the necessary functional applications. Using processes and tools, static configuration tables are generated that identify the assignment of applications to computing engines, the processor and memory resources assigned to each application, and the data movement between applications. The underlying hardware/software infrastructure controls system operations and data movement between applications using these tables.

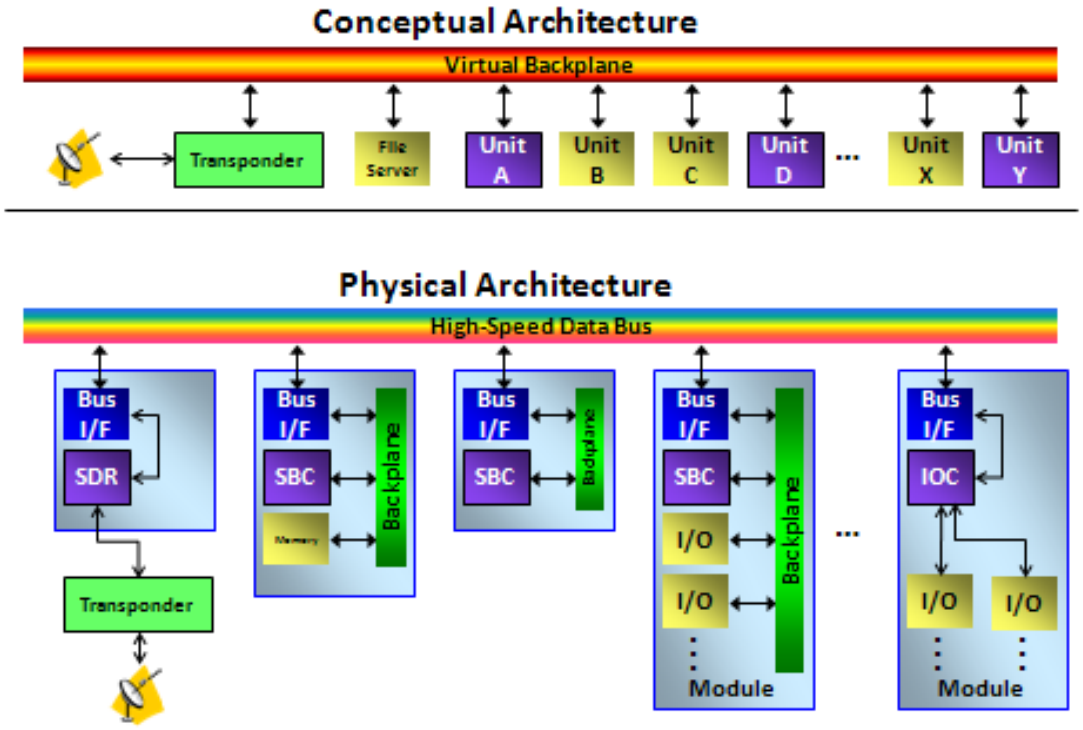

Figure 3 - *Virtual Backplane Implementation: the Virtual Backplane logically connects each unit, regardless of the physical implementation*

This architecture allows applications to be reconfigured to another element in the system by modifying the configuration tables. An application does not require modification or re-certification to be moved from one element to another. Each computing element contains a "shadow memory" which holds the memory image of the system. The unique element of the Integrated System architecture is that this "shadow memory" need only contain the subset of system data required by the applications and I/O cards associated with the computing element. This greatly reduces the amount of

memory required for a given element. Again the system remains flexible because the memory assignment is contained in the Integrated System configuration tables.

A key architectural concept known as a "Virtual Backplane" maximizes the benefits achieved from the use of common components. A virtual backplane is a method for interfacing a variable number of components in such a way that it appears they are all peers. The conceptual architecture, as previously shown in Figure 3, illustrates this scheme. Regardless of the physical architecture, all I/O data is available to any application as if the I/O interfaces were directly connected to the host processor. In the example physical architecture shown, the virtual backplane is a method for combining the High Speed Data Bus and the Module Backplane in such a way that all data is available equally to applications running on any of the Single Board Computers (SBC). This virtual backplane realization allows modules to be located at optimal sites throughout the system as needed without impacting software applications.

This architecture also allows for chassis organization to be optimally defined to reduce weight and to allow local thermal issues to be considered in the architectural implementation. Additionally, new modules can be added for extended availability, increased redundancy, and/or increased processing and/or I/O capability. This ability to add or remove modules with little impact to the existing architecture provides an approach that is easily scaled to meet program requirements. The same basic architecture can control complex vehicle avionics systems (such as the CEV), be downscaled to address a CLV approach, reduce complexity of environmental control device interfaces, or even support small and simple exploration robots.

Software and Hardware Abstraction

As described above, every function (computing, I/O, communications) appears as a peer function to any other function. This is the first key to creating an "Open Architecture" system. Since the software is a peer to the I/O (implying that there is no dependency between these two functions), an open relationship is created between the hardware and software, thus implying that different organizations can complete the hardware and software. This first "open feature" is associated with the modular nature of the architecture, a layered approach to system hardware and software provides the abstraction necessary to minimize the effect of system changes on user applications. This layered approach provides a continuous spectrum of support ranging from direct interfaces between hardware components to application program interfaces accessed directly by user applications (shown in Figure 4).

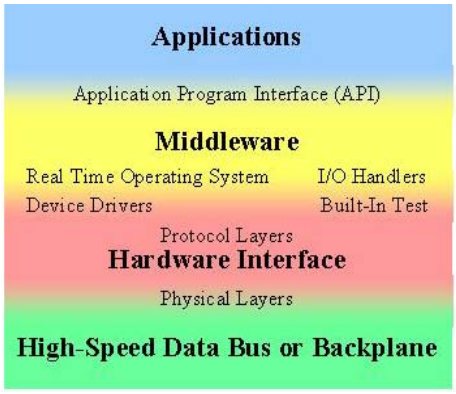

Figure 4 - *Layered Design Scheme*

Using a layered approach in the design of the bus interface controller allows the interface to the physical data bus/backplane to be separated from the remainder of the layering scheme. Migration to alternative bus architectures, even to dissimilar communications schemes such as fiber optics and wireless architectures, is simplified by isolating the impact of the migration to a minimal number of layers. This holds true throughout the spectrum. For example, changes to the operating system affect only those layers that directly interface with the Real-Time Operating System (RTOS). In general, the format of Application Program Interface (API) calls remain unchanged, isolating user applications from the affects of system modifications.

With the addition of advanced techniques in Operating System implementation, this principal can be further enhanced. Through use of an ARINC-653 compatible Operating System, the above described isolation can be extended to a single software application. This also means that each software

application can be a peer to other software applications as well as I/O and communication. Incorporation of the ARINC-653 OS into the architecture creates two open features:

- Each software application is independent and interfaces only through a "highly used open standard API call.
- The Operating System itself is an open standard and available from multiple vendors. A change from the Honeywell APEX OS to the Greenhill Integrity OS does not have any impact on any application or I/O within the system.

The realization of this flexibility is embodied within the system middleware. While this may seem a complex realization, this is work that has already been completed and is approaching 100 million hours of proven operation. This middleware has successfully been ported from the National 29050 processor in the 777 Aircraft Information Management System, to a custom enhanced 29050 version within the AIMS 2, to the IBM 750FX flight computer in the Boeing 787. This reusable middleware code is currently being ported to the 750FX on the Constellation Orion spacecraft and is the baseline implementation for the Space Suit computer. This continuous reuse demonstrates the commonality and cost savings that the NASA has been hoping to achieve.

Time and Space Partitioning

Through a combination of hardware, software, and operational tools, a single high-throughput computational platform may be partitioned into multiple virtual computers (shown in Figure 5). This partitioning occurs in four domains: memory space, computation time, I/O access, and backplane access. Each virtual computer appears as a dedicated resource to the associated software application, known as a partition. This scheme also supports multi-processing using multiple processors within the overall system. Increased future processing requirements may be implemented within virtual computers on existing processors, or on newly added processors. The real-time operating system portion of this scheme, known as ARINC-653, has gained wide acceptance recently through its use by a variety of COTS operating system vendors.

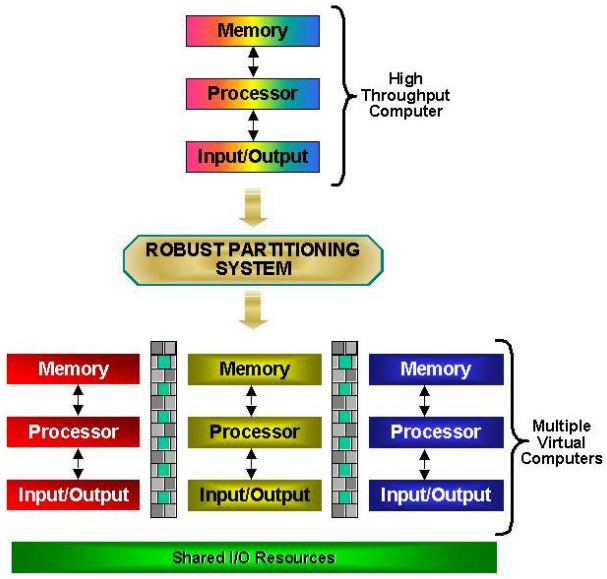

Figure 5 – *Full Time and Space Partitioning*

In the memory domain, each software partition is allocated a pre-defined range of memory resources. Based upon the needs of the software, the size and location of a partition's memory resources are allocated by the operational tools. A hardware Memory Management Unit (MMU) enforces access rights to the memory resources. Other partitions may read from allocated memory, but only the owner partition is granted write privileges. This scheme ensures that software and/or memory failures do not propagate to other partitions running on the same physical CPU. Temporary storage locations such as program registers are automatically stored by the operating system and software infrastructure when a context switch occurs.

In the computation time domain, a partition's processor resource allocation is pre-determined by operational tools, based upon the computational requirements of the partition. Using the interrupt scheme of the host SBC, the operating system/middleware performs a context switch from one partition to the next according to the pre-defined schedule. Thus a partition is guaranteed sufficient computing resources based upon the partition's execution frame rate needs. The order of execution

between partitions is consistent within each execution frame.

In the I/O access and backplane access domains, data flow is pre-determined by operational tools, based upon the needs of the various partitions. Operational tools convert the needs of various applications, along with a description of the physical architecture of the avionics system, into a set of bus/backplane access tables. These tables are used by a backplane interface controller and an I/O interface controller to control the movement of data in and out of the processor and I/O cards.

Time and Space partitioning coordinates the data flow through the system with the scheduling of processor resources available to the applications. This scheme creates a highly deterministic, high-reliability system. From the standpoint of an application, data is available in memory on the processor card when it is required, and data produced by the application is placed in local memory where the processor retrieves it for transmission to its source destination. The time-based, table-driven nature of time and space partitioning produces an environment that is conducive to easy modification, upgrade, and enhancement.

Generating a new set of tables and re-validation of the system interactions can accommodate a variety of modifications to the system. The operating system, software infrastructure, and any unaffected application source code remains unchanged.

One of the key benefits of an IMA system is that the partitioning of a computer into multiple virtual computers is seamless. Properly implemented, no partition can:

- Contaminate another's code, I/O, or data storage areas (space partitioning)
- Consume shared processor resources to the exclusion of any other partition (time partitioning)
- Consume I/O resources to the exclusion of any other partition (I/O partitioning)
- Cause adverse affects to any other partition as a result of a hardware or software failure unique to that partition

This architecture also enhances the overall processing platform reliability. A fault in a single hardware element affects only the partition(s) associated with that element. A hardware failure will not necessarily disable an entire Line-Replaceable Module (LRM). These factors allow a partition running on a single processor to be modified without requiring re-certification of other partitions running on the same processor. Thus, partitions that are subject to frequent modifications may be co-resident with relatively stable partitions without requiring superfluous re-verifications. Likewise, partitions with mixed criticality levels may be co-resident without requiring all partitions to be certified to the highest criticality level. This scheme is sufficiently mature and demonstrable that it has been certified by the Federal Aviation Administration (FAA) for commercial airlines, and by the military for a variety of aircraft.

Fault Isolation Zones

Fault handling is critical in any architecture that has the capability to adversely affect human life or mission success. In addition to compensating for simple failures, a critical computer system must account for classic Byzantine fault conditions. Table 1 is generally accepted as accurately describing the number of redundant channels required to compensate for Byzantine fault conditions.

Table 1 – *Byzantine Fault Conditions*

Required Fault Tolerance	Self Test Coverage	Cross Channel Trust	Number of Redundant Channels
0 Faults	N/A	N/A	≥1
1 Fault	100%	Truthful	≥2
	<100%	Truthful	≥3
	<100%	Lies*	≥4
2 Sequential Faults**	100%	Truthful	≥3
	<100%	Truthful	≥4
	<100%	Lies*	≥5
2 Simul-taneous Faults***	100%	Truthful	≥3
	<100%	Truthful	≥5
	<100%	Lies*	≥7

* **Classic Byzantine Fault: number of required channels is established by a formal proof**

** **1st failure removed before second failure occurs**

*** **1st failure not removed before second failure occurs**

Through use of high integrity processors, coupled to a high integrity Virtual Backplane, a master-shadow redundancy scheme, and robust BIT/BITE capability, a system having 100% fault coverage and truthful cross-channel communications can be developed. As indicated in Table 1, this combination results in the minimal number of required channels to compensate for fault conditions. There are many examples of high integrity processors within the Space community. High integrity processors can be created but are not limited to using lock-step techniques, common-monitor implementations, triple modular redundancy, and polynomial progression encoding techniques. These techniques are not new to the Space industry; lock-step processers are used in the Space Shuttle Main Engine Controller (SSMEC) and the Atlas flight computer. Polynomial progression encoding is used as the fail safe in the Shuttle Multiplexer/Demultiplexer within the shuttle avionics system.

Fault Detection and Isolation. Along with a sufficiently robust BIT/BITE capability, a high integrity step processor architecture can detect and isolate faults without requiring a cross-channel voting mechanism. Honeywell has successfully demonstrated the viability of lock-step processing using a variety of implementations. In all cases a SBC is divided in two with duplicate processors, memory, compare logic, and bus/backplane interfaces. Each side contains the logic to enable output from the other side. Both sides have to agree prior to outputting data. If the clock speed is relatively slow, all processor bus transactions, memory accesses, and bus/backplane activity can be compared on a cycle-by-cycle basis. With high-speed systems, comparing data entering and leaving the SBC is sufficient to determine faulty conditions (see Figure 6). Regardless of the specific implementation, Honeywell has used this architectural concept repeatedly to meet the FAA's requirement of <10-9 chance of an undetected failure.

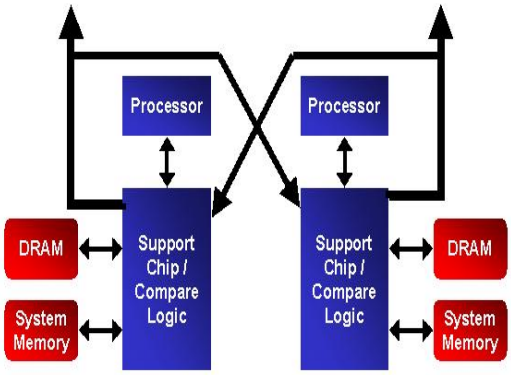

Figure 6 – *Simple Lock-step Architecture*

Fault Recovery

When the two sides of the SBC disagree, the associated LRM discontinues outputting data and attempts to correct the fault condition. If the fault can be corrected, the LRM places itself back into service upon fault correction. While the LRM is attempting to resolve the fault condition, another aspect of this integrated modular architecture assures uninterrupted system operation. This aspect of the architecture is known as master-shadowing. On a partition by partition basis (not LRM by LRM), the system architect may choose to provide one or more shadow partitions for any given partition. These shadow partitions reside on separate LRMs and receive the same inputs as the master partition. They perform the same calculations and generate the same outputs. However, the shadow partitions monitor the virtual backplane to determine if the master has provided the output data, in which case the shadow does not output duplicate data. However, if there is no data on the virtual backplane at the pre-scheduled interval, the shadow provides the output data. This scheme is repeated for as many shadow partitions as the system architect determines are necessary, assuring uninterrupted system operation. An alternate to master shadowing is to have each redundant partition place data on the Virtual Backplane at all times. Other elements then operate on the first valid data. All data will either be valid or non-existent as enforced by the high integrity processor or the high integrity Virtual backplane.

As previously discussed, current highly available COTS solutions are far less reliable than the shuttle implementation. This is because they use parallel bus implementations between their functions. Within the shuttle system, each processing element (e.g. General Purpose Computer) is connected through a serial line (Multiplex Interface Adapter Bus) to each I/O unit (MDM). Furthermore, within the MDM, each I/O card is connected to the internal controller through a separate serial line. In each of these cases, a failure in a single I/O card will not propagate and disable an entire string within the redundant system. Couple that with the fact that 64-bits of address and 32-bits of data are two magnitudes more likely to fail the system than a single serial line (or a serial pair) and the reliability difference is clear. The advent of these fault isolation zones also enable an effective failure detection and isolation capability coupled with ISHM techniques. This successful combination has resulted in the number of false failures detected to decrease from 50% in airplanes prior to the 777 to only 8% in the 777 (comparison of equal AIMS functions).

Progression to Network Node

Honeywell has observed a misconception within the NASA community relating to the IMA implementation. This misconception is that the IMA system has a "Central Computer" like the Space Shuttle and is not a distributed system. Nothing could be further than the truth. This is natural misconception to the casual observer due to the highly visible examples within the commercial community and the Orion implementation. The AIMS 777 computer is located in a centralized cabinet to facilitate maintenance of the aircraft. As part of that design requirement, the inter-cabinet bus was only designed to drive the short distance within the cabinet. In the Orion Vehicle Management Computer within the spacecraft, there are actually two high integrity processors and a low integrity communication processor located in a single chassis. This however, is not required within the architecture. The location of these three processors in a single enclosure is to facilitate the packaging at the capsule level. They could have as easily been located in six separate chassis and have been located anywhere in the vehicle. A progression of avionics concepts is shown in Figure 7. Note the colors in the progression; the central computer is all one color, indicating a single complex function. The distributed system has many boxes, each having a single function. The integrated system has boxes with many colored functions within each box. Each of the colored slices can be a single box or integrated into any numbers of chassis.

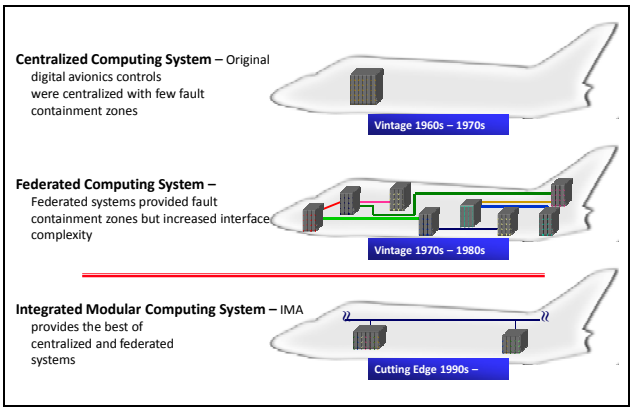

Figure 7 - *Aircraft avionics have progressed from the centralized system to the DIMA "network node" system.*

This technique is becoming prevalent within the commercial airline industry as evidenced by the findings of "Distributed IMA and DO-297: Architectural, communication and certification attributes"[2] (IEEE 10.1109/DASC.2008.4702769). This paper describes characteristics and the connection between the distributed architectural approach, its core communication system and the development and certification process. Based on the attributes of the communication system and its open interfaces, a Distributed integrated modular avionics (DIMA) architectural approach provides safety-critical and secure communication, distributed integration, hierarchical separation, partitioning and physical distribution in addition to IMA properties like flexibility.

During the implementation of the Orion avionics architecture several architectural migrations were occurring driving the Honeywell base architecture to a network node configuration. The term "network node" is intended to describe any

function within the DIMA system that can be attached to the Virtual Backplane. The Orion avionics architecture is a DIMA implementation of a network node system as shown in Figure 8. A demonstration of the Open Systems Architecture nature of the "network node" is the current activity within the current architecture. The RIU, MBSU, and ECLSS DE Units are all being redistributed and implemented as multiple Power Data Unit (PDU) assemblies and the work being redistributed throughout the Lockheed Martin Orion team.

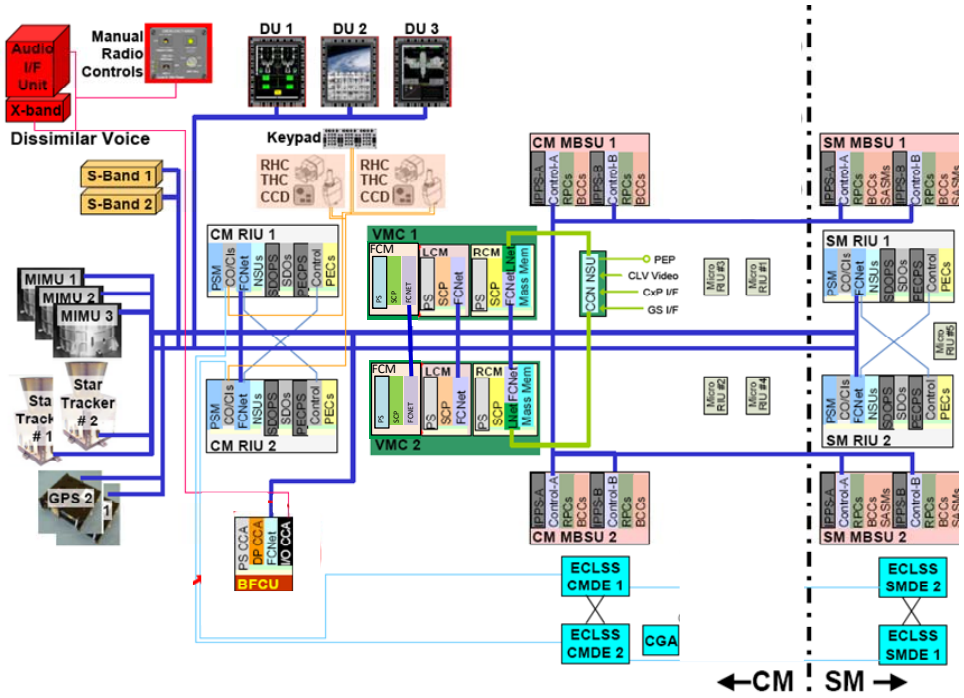

Figure 8 - *Orion 606E Baseline as published in May 2008. This demonstrates the DIMA "network node" implementation of the Orion avionics.*

TTGbE Virtual Backplane

To implement the Virtual Backplane in the Orion avionics system, Honeywell has recommended and the Lockheed Martin team along with the NASA has modified the high integrity Virtual Backplane to Time Triggered Gigabit Ethernet (TTGbE). TTEthernet, developed through a joint agreement between TTTech and Honeywell is an extension of classical Ethernet with additional services to meet time-critical, deterministic or safety-relevant conditions. It is compatible to standard IEEE 802.3 Ethernet and integrates with other Ethernet networks. As TTEthernet supports communication among applications with various real-time and safety requirements over a network, three different message types are provided:

- **Time-triggered messages** are sent over the network at predefined times and take precedence over all other message types. The occurrence, temporal delay and precision of time-triggered messages are predefined and guaranteed. The messages have as little delay on the network as possible and their temporal precision is as accurate as necessary.

- **Rate-constrained messages** are used for applications with less stringent determinism and real-time requirements. These messages guarantee that bandwidth is predefined for each application and delays and temporal deviations have defined limits. Rate-constrained message types are compatible with AFDX.

- **Best-effort messages** follow a method that is well-known in classical Ethernet networks. There is no guarantee whether and when these messages can be transmitted, what delays occur and if messages arrive at the recipient. Best-effort messages use the remaining bandwidth of the network and have less priority than the other two types of messages.

Spacewire – a simplified point to point Virtual Backplane

As part of the open system nature of the DIMA, any part of the system can be changed out. For the Space Suit program, only two high integrity processors, two I/O cards, and a communication node are required for the system. Because of this simplicity and the need for very low power, it was decided to use a simple point to point communication in the Space Suit proposed implementation as shown in Figure 9.

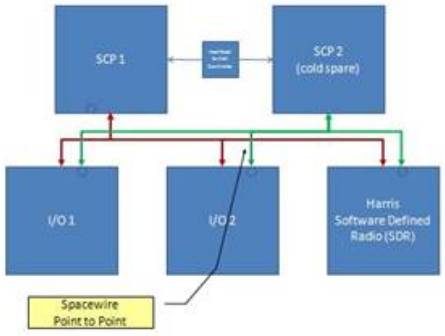

Figure 9 - *Simplified Space Wire point to point shows the open nature of the Virtual Backplane in the DIMA architecture.*

Communication Node

As noted in the Orion Design, the DIMA architecture has been expanded to include a communication node. This node utilizes a Standard Network Interface Controller (SNIC) along with a non-high integrity processing element to implement the Common Communication Adaptor (CCA) function within the Orion system. As seen in Figure 9, the CCA function is connected as a node through the point to point spacewire Virtual Backplane.

Progression to Altair and Beyond

As a progression to Altair and LSS implementation, Honeywell is making enhancements to improve the performance and make the entire DIMA system more open. First and foremost, the Altair avionics must have a much smaller Size, Weight, and Power (SWaP) footprint than is currently being realized in the Orion implementation. According to Lauri Hansen in an informal briefing, the Altair will need to be on the order of one tenth the SWaP of the Orion implementation. The current efforts 2009-2010 are designed to continue to improve performance and openness in relationship to re-configurable systems, exploration of the miniaturization of the Self-Checking Pair processor to support robotic and spacesuit applications, and to continue to explore the advancement, openness, and miniaturization of the Remote Interface Unit Controller.

Re-configurability is a system concept NASA is requiring for their Exploration Systems of which CEV is but one element. The reconfiguration goal is to demonstrate the concept of a dynamically re-configurable backplane, with autonomous configuration demonstrated with the connection of two networks.

Future space applications will require smaller and more flexible RIU designs that are fail-silent or fail-passive. An RIU design that can be readily adapted to new applications at minimal additional cost will provide advantages to NASA. This will require a controller design that provides the flexibility and throughput to handle a wide range of I/O types. Also, noting that development and qualification of software is a significant cost driver,

it is desirable to maintain a controller design that is based on hardware only or which does not require the development of custom software for each application. RIU locations used in the LDAC-1 evaluation are shown in Figure 10.

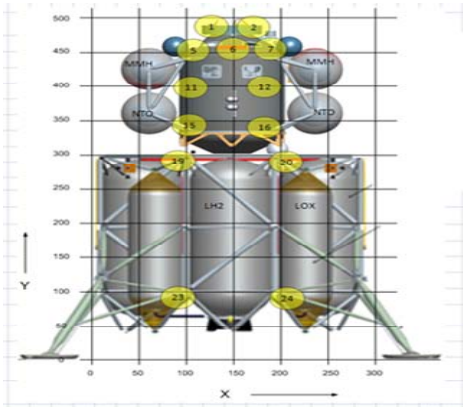

Figure 10 - *LDAC-1 RIU location used in avionics evaluation*

The Honeywell self-checking pair processor which was developed for the 787 Flight Control Module (FCM) is the basis of the Orion Vehicle Control Module and provides a radiation tolerant reliable processing element. The Constellation program has need for this kind of processing element to support robotics, lunar base facilities and spacesuit systems. The advantage of having a common processing element allows efficient use of processor modules with units being swapped to support other operations when they are no longer needed in a current mode as well as providing spare parts which are available in case of emergencies. The current miniaturized concept is shown in Figure 11. Current efforts include both Multi-Chip Module (MCM) and System on Chip (Soc) investigations.

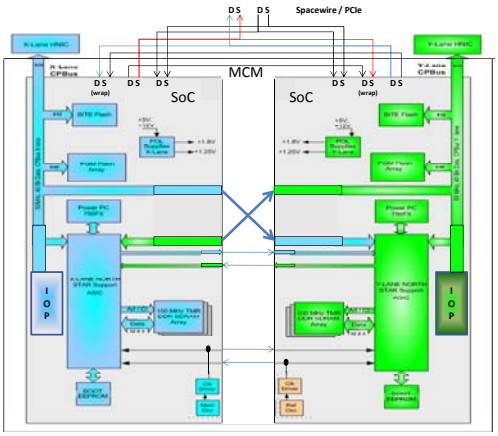

Figure 11 - *System on a chip miniaturization concept currently in work*

In addition to efforts to miniaturize the main processor, it is necessary to improve the openness and create a reconfigurable/programmable standard set of I/O intended to meet the needs of Altair and beyond. The goal is to create an RIU that is at least ¼ the size of the same RIU implemented in current technology as shown in Figure 12.

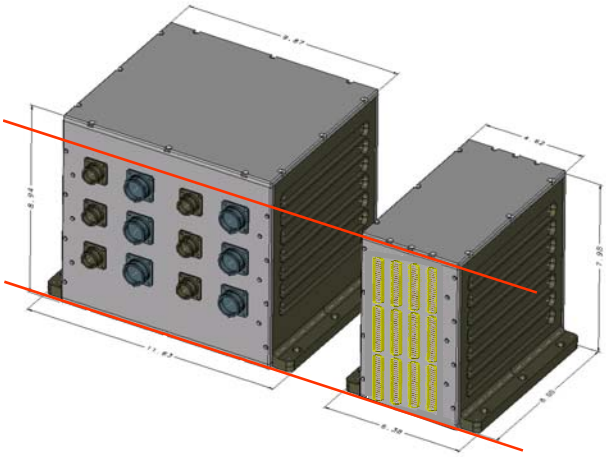

Figure 12 - *The goal is to shrink the RIU from a 6u220 form factor to a 3u160 form factor.*

- Investigate a single chip solution for the RIU Controller
- Include hooks associated with the ability to dynamically reconfigure the Virtual Backplane
- Pursue adoption of more "open" hardware architecture including adoption of the high-speed PCI-Express (PCIe) bus architecture
- Develop conceptual specs for 4 standard I/O cards that can be used for a high percentage of RIU I/O signals
  - Analog Card
  - Digital Card
  - Solenoid Card (outside scope of this effort)
  - Programmable I/O Card
- Based on such standard I/O building blocks, propose a conceptual design for a new universal RIU
- Single chip solution for Orion NIC and RIU Switch
- Investigate foundries capable of producing devices that meet space environmental requirements

Productivity and Cost

The advent of the DIMA system architecture can provide several elements of cost reduction for the NASA community, both Human Space related and extended into the satellite community. The most proven concept within the DIMA system is cost savings resulting from reduction in retest costs. Each partition in the system is stand alone and does not need recertification as the platform is upgraded. A specific example of this is that the application software from the 777 to the double speed redesigned hardware for the 777 Extended Range airplane was 98% reused without modification. The DIMA architecture also provides savings in software and integration. The full comparison of a DIMA system compared to a federated (distributed) system is shown in Figure 13.

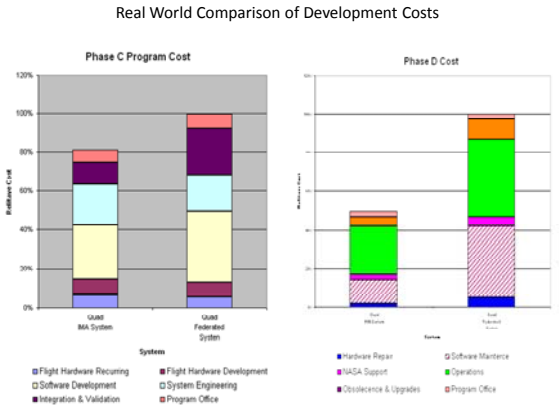

Figure 13 - *Historic comparison for DIMA costs for development and production showing the cost savings associated with advanced architectures.*

There is also cost avoidance by using common building blocks. Each element of the DIMA "network node" implementation is interchangeable (and replacable by 3rd party in the future). An example of cost savings associated with reuse of common building blocks is shown in Figure 14.

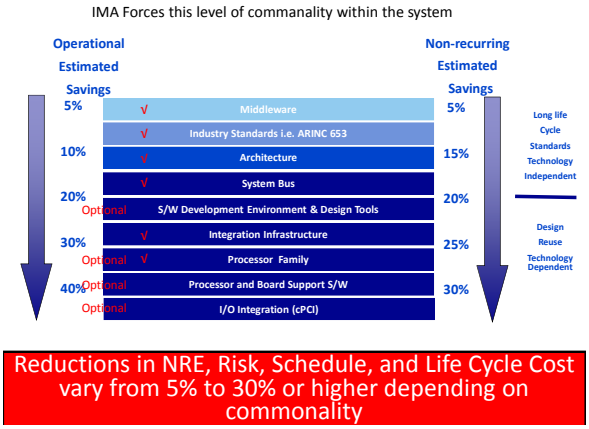

Figure 14 – *Commonality is one of the goals of an open system. Each element can be reused to reduce cost.*

References:

[1] **Open Systems Architecture - Both Boon and Bane**
Black, R.; Fletcher, M.
25th Digital Avionics Systems Conference, 2006 IEEE/AIAA
Volume , Issue , 15-19 Oct. 2006 Page(s):1 - 7
Digital Object Identifier 10.1109/DASC.2006.313746

[2] **Distributed IMA and DO-297: Architectural, communication and certification attributes**
Wolfig, R.; Jakovljevic, M.
Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th
Volume , Issue , 26-30 Oct. 2008 Page(s):1.E.4-1 - 1.E.4-10
Digital Object Identifier 10.1109/DASC.2008.4702769

E-mail Addresses

Ed Banas
Constellation Business Development
Human Space Business Segment
ed.c.banas@honeywell.com

Ralph Cacace
Constellation Business Development
Human Space Business Segment
ralph.a.cacace@honeywell.com

Mitch Fletcher
Chief Systems Engineer
Human Space Business Segment
mitch.fletcher@honeywell.com