

## A PLUG-AND-PLAY SYSTEM FOR SPACECRAFT COMPONENTS BASED ON THE USB STANDARD

Jim Lyke  
Air Force Research Laboratory  
3550 Aberdeen Ave SE, Kirtland AFB NM 87117-5776, (505)846-5812  
james.lyke@kirtland.af.mil

Scott Cannon  
Utah State University,  
Department of Computer Science, Logan UT 84322-4205

Don Fronterhouse  
Scientific Simulations Inc.,  
2951 Marina Bay Dr. #130-306, League City, TX 77573

Denise Lanza, Tony Byers  
Science Applications International Corp.  
2109 Air Park Road SE, Albuquerque, New Mexico, 87106

ABSTRACT: Plug and Play spacecraft offer the potential of simplified software development, rapid assembly and integration, latepoint addition of new components and technologies, along with more automatic testing and after-deployment flexibility. The Air Force Research Laboratory (AFRL) has established a program to develop standards for Space Plug and play Avionics (SPA) that is based around commercial technologies. In particular, "SPA-U" is based on the popular USB standard. This paper develops the basic concepts of the SPA-U standard, to include the host and client side software and hardware.

### INTRODUCTION

The term "responsive space" refers to an ability to make militarily useful space services rapidly accessible to the warfighter. Achieving responsive space requires not only new ways to build and launch spacecraft (in order to effect a far more rapid development cycle, i.e., days and weeks<sup>1</sup> instead of months and years), but also new approaches for "ordering" and receiving these space services. Current approaches to constructing spacecraft fall well short of this objective, and even with rapid launch and streamlined access to space services, the hope of responsive space<sup>2</sup> is quickly eroded without an extremely efficient way to build spacecraft.

In this paper, we propose a methodology for the rapid construction of spacecraft based on the idea that the pieces (components) of spacecraft are rapidly composable, both in design and in assembly, to form systems. We do not wish to make simpler systems; rather, we wish to make the construction of complex systems simpler through the use of automation in design, modularity in components, and standardization in interfaces, similar to the PC. Modularity, as it reflects the generic principle by which logical

boundaries in the decomposition of a system are defined, is a part of this simpler-to-construct methodology. Standardization, which attempts to guarantee repeatability in how elements are built and interfaced, also plays a part in this methodology. But modularity and standardization do not guarantee the notion that a system must be built rapidly. It is not that standards are bad *per se*. Indeed, they are essential to the methodology that will be described in this paper. It is simply the case that the blind citation of one or a hundred standards will no more guarantee rapid constructability than would random collections of standard components ensure the construction of a spacecraft or other system. Our methodology benefits from standardization, but standardization by itself will not guarantee responsive space, unless the methodology is itself encapsulated *as* a standard.

With this preamble, we develop in this paper the concepts behind our methodology, referred to as "responsivonics". Responsivonics is a collection of hardware, software, and interface concepts. The interface concepts, referred to as Space Plug-and-play (PnP) Avionics (SPA), are based on industry standards. For example, USB, which is the central emphasis of this paper, is called "SPA-U". Spacewire and Ethernet

become “SPA-S” and “SPA-E”, respectively. The reasons for this renaming will be made clear later.

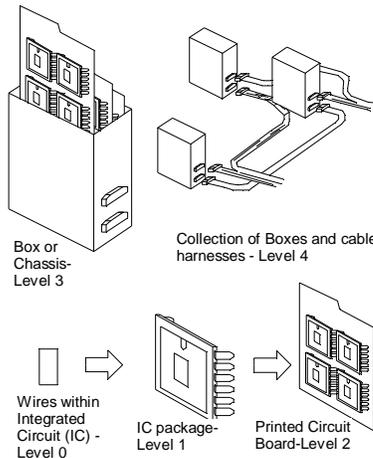
This paper is organized as follows. The next section develops the set of concepts behind the responsiveness methodology, including SPA. The multi-generation roadmap for developing responsiveness is described. Then, the key elements of SPA-U (being the front-running SPA interface technology) are described. Finally, the extension of responsiveness to other spacecraft subsystems is elaborated.

## RESPONSIVONICS

Responsivonics refers to a set of electronics concepts designed to promote the rapid integration of systems from components. We first address a philosophical view of responsiveness as impacted by the concept of modularity, using electronics as an example. From this insight, we propose an intuitional boundary from which the elements of responsiveness can be defined.

### A Rationale for Responsivonics

The packaging and interconnection of electronics systems can be viewed of as having a number of levels, the lowest being the indivisible electronic device (i.e., a transistor or resistor), and the highest being an entire platform. Packaging represents a fairly straightforward manifestation of the property of modularity. Figure 1 illustrates several hierarchical levels of interconnections present in any complex aerospace platform<sup>3</sup>. In larger scale satellite platforms, a subsystem might represent a collection of “boxes” at level 4, the satellite “bus”

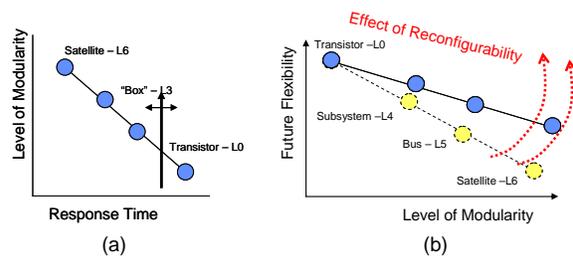


**Figure 1. The packaging hierarchy in electronics.**

might represent a “level 5” assembly, and an entire satellite might represent a “level 6” assembly (combination of bus and payload). In this case, levels 4-6 might be considered to be soft modular partitions, since they are based on collections of similar elements (i.e., “boxes”). It seems intuitively clear, all other

things being equal, that the assembly of level 4 elements (“boxes”) can be effected quicker than, say, an assembly of level 1 elements (“chips”). From this basic idea, without further justification, we claim that aggregations formed at higher levels of modularity are assembled more rapidly.

The concept is illustrated in Figure 2a, in which the level (L) of modularity is notionally plotted against “speed” or “1 / responsiveness”. In part, it can be heuristically argued that from the sheer magnitude in the number of components in systems broken at more primitive levels of modularity, it is possible to assemble a system with higher modularity (fewer components) more rapidly. The concept of flexibility can also be related to modularity. Again, we appeal to a heuristic argument. Systems broken at a lower level of modularity have more components, therefore more degrees of freedom, and more flexibility\*. This concept is roughly conveyed in Figure 2b. As another abstract concept, reconfigurability<sup>4</sup>, can be shown to have an impact on flexibility. In this context, reconfigurability is roughly equivalent to software-definability. For



**Figure 2. How modularity might be related to “responsiveness” and flexibility. (a) Higher levels of modularity are aggregated more quickly. (b) Lower levels of modularity have higher flexibility**

example, an L1 element, the *field programmable gate array* (FPGA), is a software definable collection of logic, interconnection, and memory resources based on judicious arrangements of L0 components (i.e., transistors). The FPGA is far more flexible than customized digital application-specific integrated circuits (ASICs) in which similar transistors are arranged to implement specific functional circuits. FPGAs permit the software-defined configuration of resources, whereas ASICs are less flexible because they are for the most part, not as software-definable. This concept is fundamentally important in responsive space, since in general we would opt to use elements at the

\* Simple-mindedly, all matter in the universe is based on the aggregation of atoms, demonstrating the ultimate flexibility of nature’s most fundamental of “modules”!

highest possible modular level, and we might try to enhance the flexibility of such modules by exploiting reconfigurable systems concepts.

**The Role of machine intelligence.** Careful consideration of flexibility and modularity in an avionics architecture *may* result in building blocks that can be aggregated quickly to form complex systems, but it is difficult to conceive of how this result can be insured without the help of automation. Specifically, the introduction of reconfigurable “knobs” in the interfaces of modular components could be exploited in responsivonics to maximize the likelihood that dissimilar components can be connected by reconfiguration without the need for humans to develop customized hardware / software modifications. It seems that some machine intelligence, embedded within the components, could mechanize a process in which the components being interfaced are queried and then configured automatically. We call this concept “reactive interface”, and believe it is intrinsic to most concepts of plug-and-play (PnP). Most forms of PnP that have been developed so far do not press the boundaries of this concept very far. For example, we are not aware of any PnP system involving “shapeable” FPGA resources. In this paper, we introduce a simpler concept, the provisions for component self-description through the use of transducer electronic data sheets.

### Responsivonics Building Blocks

Our strategy for responsivonics is based on exploiting modularity, reconfigurability, and standardized interfaces to achieve highly functional electronic building blocks that can be rapidly integrated through machine intelligence. In the remainder of this section,

we review four types of responsivonics building blocks. These blocks roughly represent L3 modules, with enhanced flexibility through the deliberate introduction of reconfigurability.

**Space PnP Avionics (SPA).** SPA becomes a central focus of the next section of this paper. It is a composition framework for hardware building blocks that supports flexible network topologies, fault tolerance, and dynamic configuration / reconfiguration. It also embodies the notion of machine intelligence for the purposes of accelerating integration of a network of SPA components, permitting, for the most part, an *à la carte* combination of them, to include latepoint additions as dictated by emergent mission needs. In a SPA network, a new type of threat sensor could conceivably be added immediately before launch, whose use could, in the best case, be automatically achieved through smart application software, or at least integrated dynamically into a system “registry” for *ex post facto* application code development.

**Software-Definable Instruments.** Introducing software-definable “knobs” in the design of communications equipment, guidance components, and in-vehicle health monitoring sensors (to name a few examples) emphasizes the premium placed on the information available in such components and exposes this information for general use within the system. Today’s most prevalent example is the software-definable radio (SDR). Rather than dedicating a fixed reservation of size, mass, and power for a single limited purpose in a telemetry, tracking, and control (TT&C) function, it is possible to consider the flexible re-use of the same radio frequency (RF) communications resource for other applications, ranging from

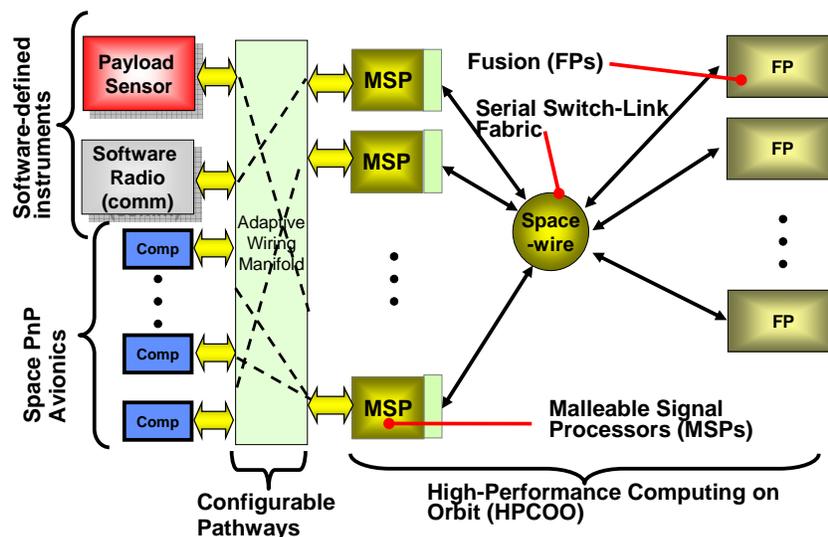


Figure 3. Avionics architecture based on responsivonics.

opportunistic relays to pseudolites and enhancements to existing missions, such as combat search and rescue (CSAR). The same resource can be used for *in situ* monitoring purposes, such as a threat warning sensor<sup>5</sup>, in which satellite anomalies might be correlated to (in this case) an RF signature, replacing fixed, dedicated resources otherwise designed for the same purpose. Beyond the SDR, many fruitful possibilities for the re-use of resources in spacecraft components can be identified, such as using imagers as star trackers (or vice versa), or exploiting multiple phenomenologies from different on-board instruments in a more correlated fashion, consistent with an *in situ* implementation of the “satellite-as-a-sensor” concept, a concept we refer to as the “self-aware satellite”. In the responsiveness concept, such software-definable instruments would exploit SPA interfaces, making it simpler both to integrate the instruments rapidly and to exploit their use in novel ways.

**Configurable Pathways.** One definition of architecture from the standpoint of electronics is the deliberate connection of components to achieve a specific purpose. The aforementioned FPGAs achieve this for digital systems by allowing arbitrary arrangements of logic and memory by soft-defining their interconnections. At the level of the aerospace platform, configurable wiring harnesses might achieve this same result. Perhaps the first configurable wiring harness concept is AFRL’s adaptive wiring manifold (AWM)<sup>15</sup>. The AWM employs an architecture similar to that used for FPGA interconnect resources, but based on the wiring demand and types associated with spacecraft. Bistable (non-volatile) and temporary (volatile) switches populate an ad hoc grid of wire crossings, programmably configured by built-in controllers. Under this scheme, a new wiring harness can be assembled from pre-built sections very rapidly, perhaps in minutes, as opposed to months for the construction of typical wiring harnesses in today’s spacecraft.

Electrical wires are but one type of configurable pathway. Conceivably, one could define reconfigurable optic, fluidic, and thermal pathways, leading to some of the more generic possibilities for PnP spacecraft described later.

**High-performance Computation On-Orbit (HPCOO).** A long-standing debate in aerospace has raged for many years regarding the utility and efficacy of processing information on-board a spacecraft, with limited resources, versus sending the same information (unprocessed) to the ground where it is possible to devote copious amounts of computation on every bit. It is more often likely in the future that turn-around time is a gating factor; the time-of-flight and latency of even powerful computers may not be adequate to enable

autonomy as an application, particularly if a platform is not in contact with a ground station. In responsive space, where it is desirable to make space resources accessible to in some cases individual end users, the directness of on-board computation is compelling. For these reasons, the definition of high-performance computation on orbit (HPCOO) is considered important to the overall framework of responsiveness.

Several important concepts are combined in HPCOO. First, the HPCOO elements are modular and scalable, combining two basic types of computing nodes with serial-link-based switch fabric approaches (such as Spacewire<sup>17</sup>). The first type of basic computing element is a reconfigurable front-end processor, referred to as a Malleable Signal Processor (MSP)<sup>18,19</sup>. MSPs are usually (but not necessarily) FPGA-based, suitable for repetitive, stream-based processing tasks that are easily (compactly) mapped to circuitry, as opposed to problems corresponding to the complexity class defined in computing science as NC-hard<sup>20</sup>, which can take exponential spatial resources (i.e., gates) to implement.

## SPACE PLUG-AND-PLAY AVIONICS (SPA)

The elements of responsiveness introduced so far emphasize the important properties of mobility, scalability, and reconfigurability. But responsiveness additionally requires mechanisms that intrinsically promote rapid assembly, integration, and test. The *space plug-and-play avionics* (SPA) concept defines a particular set of interface-driven concepts that are structured specifically to address this objective. The interfaces between components, as logical boundaries, represent a natural “target of opportunity,” and the SPA aims to achieve this through *pure* modularity, to mean glueless hardware and software modularity. “Glueless” is a very constrained form of modularity that allows rapid integration to occur. In the construction of modular spacecraft, we often find that custom interface circuits and software must be developed as the components are brought together to form a more complex assembly. So, even with traditional modular approaches, it is necessary to “add glue”. Glue is the hardware and software needed to logically bind together components, even modular ones, to form a coherent system. Glue, unlike modular components, has an unknown complexity that complicates the otherwise straightforward notion that components can be simply plugged together to achieve this result. The uncertainties of hierarchical aggregations of such assemblies (along with their associated glue hardware and software) create a cascade of glue, and we assert that the need to account for this customized glue creates an intuitive expectation of

delays in the integration process. As such, development and flight schedules codify these delays as standard practice. By contrast, in SPA, we seek engineering approaches that by design eliminate the need for “glue”. We believe this can be achieved by a combination of inserting reconfigurable resources (e.g. circuitry) in the interfaces of components and automating the settings of these reconfigurable resources. This amounts to “morphable glue” (motivated by the concept of reactive interface mentioned previously), which in principle reduces the integration of systems to plugging operations, accompanied by highly automated (console driven) test and verification protocols.

One of the most exciting prospects of the SPA approach is that many human-induced errors in interpretation may be eliminated. At least some of the manually interpreted interface control documents will be replaced by machine-readable electronic equivalents. In SPA, every component carries an electronic datasheet whose contents automatically negotiate with the host into which it is plugged. There is an important ramification in this idea of machine-negotiable interfaces. Rather than simply investing the silicon [gates] in embedded designs for the end goal of achieving some raw performance result, we now must divert some of this silicon into the interface itself for the auxiliary goal of accelerating the pace of integration. The idea of shifting part of the functional capacity of computing into interface is not a new idea, but is evidenced in every personal computer, where human productivity benefits from the investment of what at one time would have been considered a profligate amount of computing machine cycles and memory resources. As a result, most humans today are mercifully spared the need to work with assembly language and crude console interfaces, whereas in the past availability of computing resources were too scarce to justify simpler user interfaces. In the 1990’s the PC industry struggled to create a concept for PnP<sup>21</sup> technology to simplify the integration of components and promote widespread availability of third party components. Here, too, an initially significant overhead was introduced. But, with the knowledge that Moore’s law would in time reduce any fixed quantal block of gates to an arbitrarily small size, power, and cost, the industry introduced standards such as USB and PnP application program interfaces and drivers that eventually fulfilled those notions of low cost and abundant availability of components. As such, we indicate that SPA, like other initiatives in interface simplification, will require a certain investment in overhead initially for a longer term benefit, in this case the profound reduction in the time necessary to construct and field a spacecraft.

### ***The Traps of Building a Better Mouse***

The SPA concepts are inspired by and benefit from the significant investment of research and development that led to, in essence a better mouse (and better keyboard, etc) by virtue of a reasonably flexible interface. Why not simply use the same infrastructure – if it is successful – to produce the same effect in the spacecraft industry? Certainly, one consequence of the vastly difficult economics in the aerospace and PC industries drives part of the answer. In the case of USB, real time operating systems have limited support for standard software drivers. The lack of prevailing standards in spacecraft command and data handling (C&DH) processors makes the prospect of developing software drivers impractical. Most operating systems that support PnP are incapable of meeting hard real time constraints needed in embedded systems. Furthermore, the excellent power management facilities available within the USB standard cannot be directly applied to spacecraft components due to their higher power consumption as compared to consumer PC components. The lack of availability of radiation-hardened components introduces additional complications, as it is necessary to recreate interface components in radiation-hardened form. Recasting electronic components in radiation-hardened form poses a number of challenges, the most basic of which involve the choice of complying with a commercial standard (to include any overhead functions, even those unnecessary to a real-time embedded system) or producing an unsupported variant. Often, even if the standard interfaces can be faithfully replicated in a radiation-hardened form, the multi-generational performance gap between rad-hard and commercial processes can result in a relatively cumbersome and expensive implementation of an otherwise simple and efficient interface. As such, the movement to PnP by simply adopting the most common commercial approaches becomes a most unattractive proposition.

### ***Foundational Principles for Space Plug-and-play Avionics (SPA)***

Considering the difficulties in the well-meaning migration of commercial technologies to a form suitable for most space systems, AFRL researched a number of PnP-like networking approaches. These included: schemes based on the automatic recognition of nodes in a distributed system based on fault-tolerant operating systems<sup>6</sup>, redundant multi-drop networks<sup>7</sup>, and dynamically-scheduled networks (i.e., our “liquid manifold” concept<sup>8</sup>), and alternate interconnection approaches (e.g., through superimposing communication on power distribution networks<sup>9</sup>). The insights gained from a variety of these research projects led to the pursuit of a universal strategy for networks of

rapidly-integrable components. The properties for implementations of this approach, which we originally called appliqué or “peel-and-stick” sensors and later renamed to SPA, are formulated as follows:

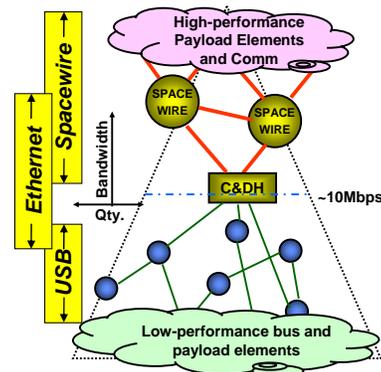
- *Light-weightedness.* PnP clients / devices should have the simplest possible interface, as measured by gate-count and lines of code.
- *Latency.* Information must be sent or received in the shortest time possible.
- *Bandwidth.* The network should support maximal information transport from a single device and from the largest collection of devices.
- *Scalability.* The number of devices that can be connected should be very high.
- *Robustness.* The interface circuits and networks should be resilient to faults and degradation imposed by the space environment.
- *Completeness.* The PnP concept should be general enough to accommodate the widest conceivable diversity of device types. It should not be necessary to re-engineer the PnP concept when a new type of device (spacecraft component) is introduced.
- *Rapid Integrability.* The PnP network should support the addition and removal of devices without the need to modify wiring and software. If the PnP device is sufficiently complex, additional software may be needed to properly exploit it, but to the degree possible, the network should be able to “handle” the device (power, test, telemetry, and registration) in a predictable default manner, permitting late-point addition of application code.
- *Existence of Infrastructure.* The PnP network should be able to accommodate a wide range of existing interface standards. Existing standards enjoy the ready availability of intellectual property (IP) for building radiation-hardened versions of interface circuits and a wide range of development aids, such as protocol analyzers and available expertise.
- *Support for peer-to-peer networking.* We believe that centralization creates additional complexity, and that a centralized C&DH processor is unnecessary. While the functions performed by the C&DH processor are important, the functions need not be performed by a single central processor. We believe that networks developed to support decentralization will ultimately be simpler, more supportable, and more robust.

- *Automated join, discovery, and ease of use.* The infrastructure for automating PnP includes the ability for nodes to self-register to networks. PnP devices should also be self-describing, and the mechanisms for managing single or networks of PnP devices should be as simple as possible. Applications should be reducible to a series of primitive transactions between PnP objects, consistent with these expected mechanisms of self-description and self-registration.

These properties drove a few postulates. First, we argued that to be as simple as possible, PnP devices should support a single-connector interface, containing the electrical conductors necessary for signal and power distribution. This idea follows the approach of USB, firewire, and power-on-ethernet. Next, to support the automatic description of services, we examined the use of embedded datasheets, following the example of the IEEE 1451 series of standards<sup>10</sup>. The types of information contained in this datasheet would at a minimum include a description of “services” and “knobs”, but could be extended to include a wide range of abstract information such as geographic position, user’s manuals, and maintenance history. A third postulate is that PnP devices have an automated means of discovery by the network. This property is consistent with most commercial PnP concepts, including Jini<sup>13</sup>, universal PnP<sup>11</sup>, and Salutation<sup>14</sup>. Application software should be disciplined, composable based solely upon the services described in the embedded datasheet. Finally, we argue for support of fault-tolerance, dynamic network changes, and redundancy to promote resilience in PnP networks.

### Choice of Interconnection for SPA Interface

Under a recently established committee on standards



**Figure 4. Bandwidth supply and demand.**

(CoS) approved by AIAA, AFRL is leading a team to implement SPA interfaces based on USB, Spacewire, and Ethernet standards. The choices for these particular interconnection approaches are briefly explained, and

the USB-based implementation is described in further detail.

First, we must explain why the choice of more than one interconnect approach is necessary. As notionally depicted in Figure 4, spacecraft components typically represent low bandwidth demand and payload components are high demand. The figure overlays the communication bandwidths of USB, ethernet, and Spacewire. Some interpretation has been applied to this representation. For example, while USB 2.0 is capable of support transfer rates above 400 Mbps, USB networks operate as dynamic multidrop busses, meaning that this bandwidth is shared amongst all devices on a single network. As such, intermingling many components at disparate communications rates could tax the limited available bandwidth of that bus structure. While spacewire, combined with intelligent routers, forms a flexible switch fabric, the mixture of very high and very low performance devices can produce a situation analogous to mixing dirt roads with superhighways. Furthermore, spacewire (as well as Ethernet) are not very lightweight as measured by gatecount. Given the multi-generational lag in radiation-hardened electronics compared to commercial technologies, the amount of overhead is very high, making it difficult, for example, to justify the addition of an ethernet link to a single thermometer. For these reasons, we have pursued USB-based SPA for low-bandwidth devices (arbitrarily chosen as < 10 Mbps), and Spacewire-based SPA for higher-bandwidth devices (> 10 Mbps).

Ethernet (100 Mbps), which is also being examined for SPA implementation, would overlap the spans covered by USB and Spacewire. As in terrestrial systems, the pressure to apply ethernet is ever-present, and we are aware of a number of aerospace research programs that are considering it as an interface, if for no other reason than its ubiquity.

Other interconnection standards, such as Firewire and RapidIO, are being considered for future implementation as SPA standards. It is recognized that the 625 Mbps performance limit of our current implementation of spacewire is not high enough for demanding payloads. At the same time, we seek lighter weight interconnection systems that embody the properties previously outlined. For example, a lightweight version of spacewire has been developed that does not support the highest transport rates but occupies a small fraction of the gate complexity of the original interface as defined by the existing standard.

### ***Self-Description of Components and PnP Middleware***

In the PC form of PnP technology, drivers are pieces of software resident on a host machine that enable the PC to communicate with a device. Drivers in this case are specific to an operating system (OS). The lack of a universal OS for command and data handling (C&DH) systems makes design of drivers problematic, since every conceivable combination of spacecraft component and C&DH system would need to be supported. As such, “driverlessness” becomes an important aspiration for SPA. Two concepts developed as part of the SPA approach support driver-less design: the XML-based Transducer Electronic Datasheet (xTEDS) and a middle-ware approach known as the Satellite Data Model.

**xTEDS.** While the USB standard supports PnP or automated enumeration of avionics devices in a network, the xTEDS concept supports PnP recognition of the capabilities, resources, and data products of an avionics device. An xTEDS is a simple XML document that describes a device to a PnP network. Each spacecraft component becomes a “SPA device” that is associated with an xTEDS contained within the component. Embedded within an xTEDS is a description of device controls, command messages accepted, variables produced, and data messages that can be delivered. The xTEDS concept is both a document and a protocol. As a part of PnP, a newly-attached device is automatically queried for its xTEDS and this information is registered with the system.

As such, each SPA device must support a particular interconnection standard (e.g., USB), but also must be capable of supporting a messaging protocol for data delivery, control, and utilization as defined within its particular xTEDS. The combination of the xTEDS and USB protocols provides the ability for component self-description: when a SPA device is attached to the SPA network, it is first recognized and enumerated via the USB protocol. Next, the xTEDS protocol allows registration of all device capabilities and data products, allowing the device to describe itself to potential users and subsequently respond to their commands and requests.

### ***Satellite Data Model (SDM).***

It is possible to view SPA as a layered concept, similar to well-known open systems interconnection (OSI) model<sup>12</sup>, as suggested in Figure 5. In this vertical model, the actual SPA devices/components occupy the lowest level, including the interconnection transport (i.e., USB). The Satellite Data Model (SDM) is a middleware concept envisioned as a distributed software program. The SDM allows applications to

coordinate, share data, find resources, and provide resources and services without being programmed to know physical location and messaging structure of other system processors or sensors. This is accomplished through the previously-described xTEDS information registered by other processes and sensors within the SDM. Applications are written to interface to the SDM. The discipline of application design under SPA involves use of a mechanism to query the SDM for registered xTEDS information and to request data messages from these other processes and sensors. Applications are also provided a mechanism to register their own xTEDS information in order to provide data and services to other applications. The design of an overall mission, itself an application, can also be viewed as another layer in this vertical view of SPA.

There are five primary “managers” (specialized software executive programs) defined by the SDM: *processor manager*, which resident on each processor and responsible for keeping that processor busy; *data manager*, which keeps track of all data available at any given time; *task manager*, which keeps track of active and pending tasks; *sensor manager*, which provides an interface between most SPA components to the processing network; and a *network manager*, which explores the network and maintain routing tables. These managers are logically a single function, even though they can have a multi-instantiated distributed implementation.

**Data Manager.** The Data Manager is a key component of the SDM. It is the focal point for process (device and algorithm) resource (data, commands, and services) registration. The Data Manager uses this to maintain the Data List which contains all of the resources available to all users. These resources can represent either physical devices (e.g. sensors and actuators) or computational process (e.g. algorithms). In addition, to help reduce data clutter, the Data Manager maintains a list of messages (defined group of variables) that are in current use. A data request is first satisfied by an existing message if possible. The Data List and Message List are updated / pruned as processes are added or terminated.

The Data Manager also handles processes requests for data. A process sends either a request or a query to the Data Manager to receive a data item. Requests can be for a specific data variable or data message (defined group of variables). If a data request cannot be satisfied from existing data sources, the Data Manager checks the Process Library (see the next section on the Task Manager) to see if a process is available the data request. If so, a task is added to the Task List to execute that process. Alternatively, if the data can be supplied by a device that is turned off, a request is made

to turn it on. This request obviously must be vetted by mission processes. If a request cannot be satisfied, the requesting process is notified.

Queries are very powerful tools and can request information by variable, message, device type, etc. This means that a process can utilize many sources of data. For example, if a primary data source for a process fails, a query for alternate data sources can be made. If suitable alternate data sources are found, the data can be requested and the process continues. This intelligent use of alternative data sources is a primary benefit to a data oriented model.

**Task Manager.** The Task Manager is responsible for maintaining the system of processes that are required at any particular time for the active mission. It keeps track of which processes are running on which processors. It keeps a list of all pending tasks via a Task List that can be queried by individual Processor Managers in search of tasks to execute. Tasks can be placed in the Task List by the Data Manager or other processes.

The Task Manager also maintains the current list of available processes that can be executed in a dynamic Process Library. The library (which can be updated with new modules after launch) contains a module id, a module name, the resources required to execute, CPU type, amount of memory, amount of CPU cycles, any special resources, data produced (short list of just names from the Common Data Dictionary), and a pointer to the executable. This should be enough information for a Processor Manager to evaluate if it can execute a specific task and for the Data Manager to assess if it can fulfill a pending data request.

**Processor Manager.** The Processor Manager is charged with keeping its processor utilization within a predetermined range. It also provides basic services to processes such as interrupt handling, multitasking, module loading, and messaging. When CPU utilization is low, it retrieves the list of pending tasks from the Task Manager and then offers to execute those that match its current resources. If the processor is selected to execute one of the tasks, the Task Manager returns the task module.

The Processor Manager implements inter-process messaging. It keeps the individual message destination table provided by the Data Manager along with the valid input message table. By monitoring the message activity, the Processor Manager determines when a process is idle.

The Processor Manager also handles process termination based upon process specific termination conditions. These can include time process is idle or a termination message. The Processor Manager also handles system control messages. These include the heartbeat regularly sent to the Task Manager with the processor status, system mode messages, and process termination messages.

**Sensor Manager.** The Sensor Manager is responsible for interfacing a specific data network to the SDM. Usually these data networks are groups of devices (sensors and actuators) with limited compute and messaging capability. To support the self defining ICD, the devices utilize an xTEDS, described earlier, that describes the data produced or used and the commands responded to. There can be as many Sensor Managers as required for a specific system based upon topology (how the devices are interconnected) or interconnect type (e.g, USB, Ethernet, SpaceWire, etc.)

As a device is detected, the Sensor Manager queries the device for its xTEDS and then creates data registration messages for the Data Manager. The Sensor Manager then receives messages sent to specific devices on the network it manages and translates those messages into network specific messages and delivers them to the device.

**Network Manager.** The SDM assumes the satellite is a heterogeneous mixture of computing elements that are

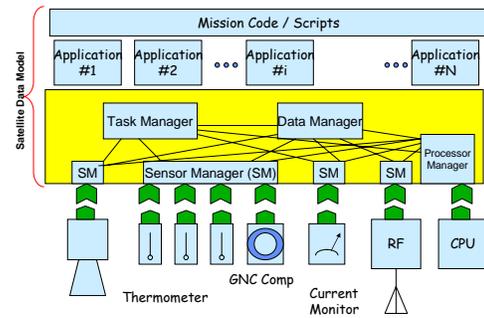


Figure 5. Vertically-layered view of SPA.

### Test Bypass Interface

In recognition of the significant complexities in integration, even with a PnP technology, we are developing concepts for a test bypass infrastructure, which permits the direct and uniform connection of SPA components to a hardware-in-the-loop simulation (HWILS) apparatus. Ideally, test bypass is implemented non-invasively, permitting the spacecraft to operate realistically in a ground test from the viewpoint of avionics processing and data flow. A simple example, shown in Figure 6, motivates the need for an improved ground test concept and how test bypass might possibly address this need. In this example a simple SPA device, a thermometer, is connected to a SPA interface (Figure 6a), in this case a

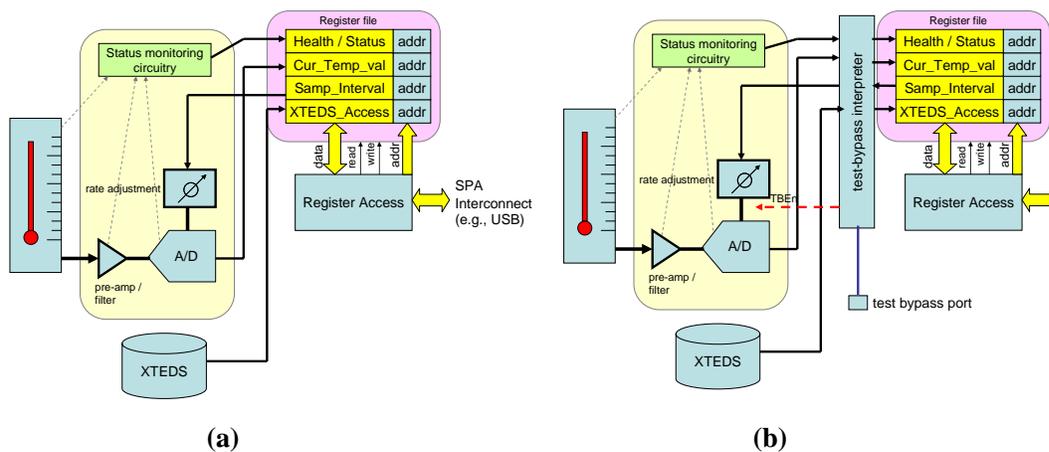


Figure 6. Simplified depiction of test bypass. (a) Register file model. (b) Incorporation of test bypass.

interconnected with efficient messaging. The Network Manager is an agent that determines the location of all computing elements (including processors, routers, sensors, actuators, etc.) that are addressable on the network. A Routing Table from all to all is maintained to support the efficient addressing of messages.

USB-based (SPA-U) interface. The temperature and thermometer status, as well as the thermometer sampling rate, are mapped to a simple data register structure, abstracted ultimately through the xTEDS description for the device. Presumably, a bidirectional interface exists to permit query (reading of temperature and status) and manipulation (writing the sampling rate)

of the register structure contents through the USB interface.

It is a simple matter to test the thermometer in a generic sense by supplying heat. The SPA device would reflect the variations in temperature in an expected way, accessible through the register structure using the SPA interface. In a complex system, however, containing many such thermometers it is a more difficult undertaking, particularly if it is desirable to have the different thermometers reflect controlled settings on mission-like timescales (i.e., the so-called “day in the life” simulation). But, especially in simple sensors, the application of heat is a non-remarkable occurrence, probably not worthy of the significant investment for developing a more elaborate vectored precision temperature generation system. Chances are high that simply inserted the numeric quantity equivalent to temperature would suffice for the purposes of system integration.

It is for such cases, which we believe constitute a great many useful instances in systems test, that the test bypass concept was devised in concert with the SPA approach. Now, the example SPA thermometer device, shown in Figure 6b, is modified to accommodate test bypass through the addition of an external interface and interposing logic system. The logic system, or “test bypass engine” intercepts the register file structure, enabling the manipulation of the contents by an external connection. In this manner, the temperature provided to the SPA network can be replaced synthetically. For the cases where the component is assumed to be good, test bypassing provides an effective means of orchestrating the actions of a large number of spacecraft components through a simulation interface.

## SPA-U IMPLEMENTATION

There are many properties of the USB interface desirable in the construction of a PnP approach. The USB interface enjoys significant commercial support and ubiquity, with intellectual property (IP), commercial protocol analyzers, and a large body of available expertise. It is a reasonably lightweight implementation, consistent with its use in PC keyboards and mice. Even the slower full-speed USB supports 12 Mbps maximum transfer, far greater than the MIL-STD-1553B 1Mbps transfer rate. We would not use USB for the primary data transport in high-performance payloads, but for most spacecraft components, full-speed USB provides ample headroom for command and data handling. The connection topology of USB is structured as a multidrop bus, but connected as a dynamic, directed tree in which the leaf nodes (i.e. components like keyboards and mice) are called *endpoints*, the multiple-ported nodes are called *hubs*, and the root node is called the *host*. A single USB tree

can have as many as 127 endpoints, and this limit can be overcome by employing multiple USB networks in a single system. USB networks are reasonably robust, supporting the dynamic (hot-swapped) addition and removal of devices in very flexible topologies (the exact network location of endpoints is not important). The USB connection is both complete (provides power in many cases adequate for devices to operate, as well as control and data transport) and rapidly integrable through a single plugging action. The process of enumeration is intrinsic to the operation of USB, in which devices are joined to a network, discovered by the host, and then matched to an appropriate driver.

For all these advantages, USB has a number of limitations that would be desirable to overcome for aerospace applications. The most significant drawback is the lack of radiation-hardened components. Another significant limitation is the lack of precise synchronization between components and host. Though USB has a very rich power management facility, it is “underpowered” (i.e., 500 mA at 5V) for most space devices. In the software design for traditional PC OS environments, drivers resident on the host machine are matched to devices. For SPA, this need to match drivers to components would result a severe limitation to rapid integrability due to the lack of standardization in C&DH hardware. Simpler barriers to USB usage in space systems include the fragility of USB connectors in the anticipated launch vibration and thermal environment.

Our goal in formulating a SPA-U concept was to preserve the benefits of a widely accepted standard (without compromising the core design of the standard), while enhancing the facilities of USB to provide additional robustness, power-handling, and synchronization.

### *SPA-U Pin-out / Connectors*

The current definition of SPA-U is a nine-conductor pin-out:

- Pins 1-4 are the pins corresponding to the current USB pinset ( $V_{BUS}$ , D-, D+, Ground)
- Pins 5-6 are pins for 28VDC power (at up to 3A) ( $V+$ ,  $V-$ )
- Pins 7-8 are pins for a 3.3V level, RS-422 interface implementing to a 1PPS synchronization signal ( $S+$ ,  $S-$ );
- Pin 9 is a pin for a single point ground (usually chassis ground) (SPG).

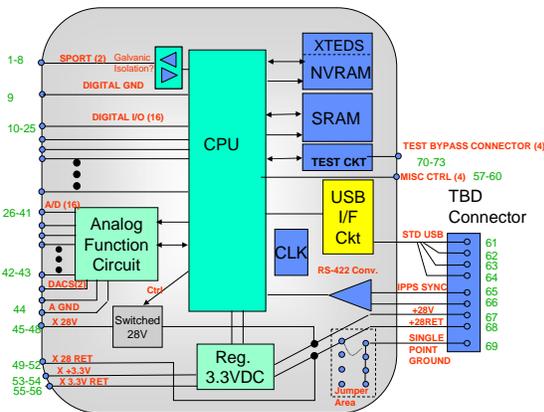
Connector styles for SPA-U under current consideration include the MIL-DTL-85313 (Micro-D) and DSCC

94031-94046<sup>†</sup>, though for near-term prototype work the classic nine-pin “D-shell” has been used with the convention that all components, including host and hub components, will employ female connectors.

### Endpoint Design

Endpoints that support SPA-U interfaces are called SPA-U devices. SPA-U devices support a single primary SPA-U connection and a secondary test bypass connection. To support PnP operation, SPA-U devices contain xTEDS defining at least one particular service and support a simple communications protocol superimposed on USB. SPA-U devices are expected to use the SPA-U connector for primary power and synchronization (if synchronization is required).

**Applique Sensor Interface Module (ASIM).** While it is possible to construct custom circuitry to support these requirements, a significant part of the SPA-U initiative is working to establish reference implementations that bundle the support circuitry necessary to implement the construction or conversion of most spacecraft components to the SPA-U format. This circuitry is referred to as the ASIM. Ideally, the ASIM would be a small embeddable module<sup>16</sup>, designed to resemble an integrated circuit package. The original concept for an ASIM design based on the SPA-U interface is shown in Figure 7. The essential elements include: a self-



**Figure 7. Original ASIM concept.**

contained microprocessor, USB interface, switchable 28V(+/- 6V)DC, a test circuit (“test bypass engine”), self-contained memory for program and data (to include the resident xTEDS), and synchronization circuitry. Additionally, user “facilities” would be provided in an idealized ASIM, to include programmable digital discrete bidirectional input/output signals (I/O), analog I/O, serial communications port(s), and accessory power (e.g. 5V, 3.3V). The user would write application code using a built-in ASIM library

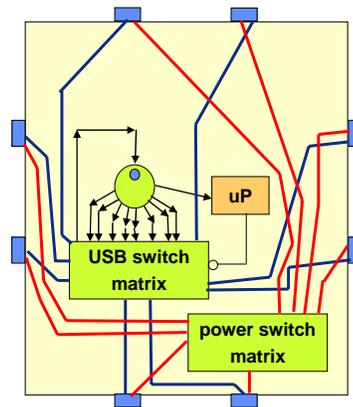
<sup>†</sup> Connector recommendations courtesy of John DiPalma, Spaceworks, Inc.

containing application programming interface (API) calls for common routines, freeing the need for users to delve into low-level SPA-U interface management.

**ASIM Device Protocol.** SPA devices, in particular SPA-U ASIMs, must be capable of decoding USB messages according to a simplified messaging format consisting of three (3) fields: *Command*, *Length*, and *Data*. The first byte is a Command token, consisting of a single upper-case ASCII character. The Length field consists of a 16-bit (2 byte) unsigned integer representing the number of data bytes. Data then represents a sequence of raw byte values. The maximum length of a command message (from the host) is 64 bytes. The maximum length of a response message (from the ASIM) is 16k bytes. The length field is ordered as having its most-significant-byte-first.

### SPA-U Hub Design

To enhance the robustness of the USB hub concept, the SPA team has developed a “robust hub” technology. The high-level hub design, shown in Figure 8, combines an unmodified USB hub IP with other components to form a self-orienting multi-ported USB



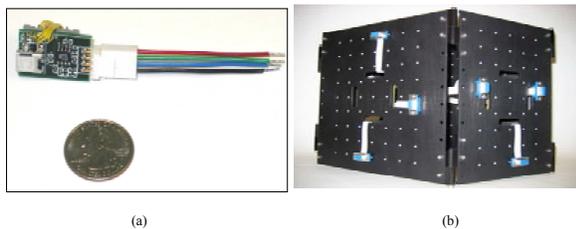
**Figure 8. SPA-U hub concept.**

hub. In principle, any external port can be mapped to any port of a standard USB hub (i.e., the type used in PC networks) through an analog USB switch matrix under control of a built-in (captive) microcontroller ( $\mu$ P). As stated before, USB is traditionally a directed tree network. In standard USB hubs, one port is distinguished as an *upstream* (connects to host), and all other ports are *downstreams*. In the SPA-U hub, all ports are potentially either upstream or downstream, though only one upstream exists at a particular time. This single upstream is selected upon initialization by the hub’s captive microcontroller. It connects the hub’s own upstream (see dot on hub, shown as a circle in Figure 8) to the first detected upstream found on any external port. Power management for the 28V connections is handled by separate relay circuitry, also

controlled from the captive microcontroller. The microcontroller itself is derived from the ASIM design, and the SPA-U hub is therefore also a SPA-U device, complete with xTEDS.

## STATUS OF SPA IMPLEMENTATION

Five SPA workshops have been held since July 2004, with numerous splinter working groups. The AIAA has approved a committee on standards (CoS), and a technical committee has been formed to create SPA standards and guidelines. Four working groups are currently defined: Generation 0 (COTS implementation of SPA-U ASIM and hub), Generation 1 (rad-hard



**Figure 10. Early SPA-U work. (a) Partial ASIM prototype based on USB-enhanced 8051. (b) Panel configurations to study hub / endpoint interaction.**

implementations of SPA-U, SPA-S, and SPA-E (or Ethernet-based SPA), Software (overseeing development of the SDM and xTEDS ontology), and Responsivonics (looking beyond the current set of SPA technologies). Supporting all of these developments is an AFRL initiative to develop a responsive space testbed (RST), which provides an infrastructure for validating the concepts of SPA. Summary to date is provided roughly by working group.

### Generation 0 Activities to Date

The first ASIM-like prototypes of a prospective SPA-U were created in late August 2004 (Figure 10a). This simple circuit was based on the Cygnal C8051F320 USB-enhanced microcontroller. At that time, the xTEDS facilities were accommodated by allocating a portion of the small non-volatile memory space, and the ASIM did not support power management and synchronization. Nevertheless, a quantity of these simple ASIMs were built and used to demonstrate PnP networks (the host was implemented on a Linux computer in which the driver software was modified) based on very simple in-house sensors, including a GPS receiver and magnetometer. In January 2005, the prototype of a self-orienting hub was demonstrated. The combination of these simple ASIMs and hubs were subsequently operated using a number of mock spacecraft panels, shown in Figure 10b. These early demonstrations proved feasibility of component

addition, removal, and relocation under dynamic operation of the simulated host C&DH processor.

Radiation tests (test configuration shown in Figure 9) were performed on some of the components (Cygnal processor, TI USB hub, Linear Technology switch, and National switching regulator) used in these early endpoint / hub experiments at Indiana University Cyclotron Facility. The results of these tests revealed no transients, upsets or latchup at energy levels of 200 MeV total fluences at or in excess of  $1.4 \times 10^{11}$  P/cm<sup>2</sup>



**Figure 9. Test configurations used in proton testing.**

for the hub, switch, and regulator. However, the commercial processor did experience latchups, which did not appear to be destructive. With the exception of the Cygnal processor, these results, though not conclusive, are an encouraging indication for suitability for use in space environments. It may be possible to implement rapid recovery approaches (e.g., external watchdog timers) on the Cygnal, if the latchup modes can be established as non-destructive and warm recovery of the processor is fast enough to avoid disruption in operation.

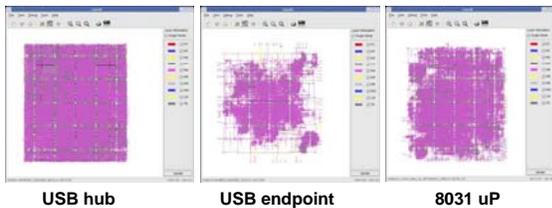
Currently, a more complete SPA-U ASIM (for use in endpoint designs) and SPA-U hub are under construction, with potential availability to third party users as early as September 2005. These ASIMs will be physically larger than a desired flight configuration, but will be useful for early development. More compact “Gen 0” SPA-U ASIMs and hubs ( $\leq 8$  in<sup>2</sup> target) are expected by April 2006.

AFRL is investigating near-term demonstrations of Gen 0 SPA-U components on sounding rockets as well as low-earth orbiting spacecraft. It is becoming clear in the planning of these projects that a version of the SPA-U hub with enhanced power-handling will be useful. One of the configurations under investigation will have two “SPA-UH” ports (“H” meaning “high-power”) and four “normal” SPA-U ports.

### Generation I Activities to Date

The primary difference in the emphasis of “Gen1” compared to “Gen0” is in developing IP that is radiation-hardenable through synthesis in radiation-hardened processes. In Gen1, the range of activities are also expanded beyond developing rad-hard SPA-U components to include the development of SPA-S routers and link-endpoints.

Commercial IP for a set of USB components (hub, endpoint, and host) were procured late 2004, along with the IP for a generic 8031  $\mu$ P. The endpoint IP was configured to support direct interface to the 8031  $\mu$ P; otherwise, no modifications were performed on the IP blocks. Using an experimental structured application-specific integrated circuit (ASIC) technology under development through an AFRL-managed DARPA program, these cores (shown in Figure 11) were synthesized and fabricated in a 0.13 $\mu$ m technology as digital-only test die. The components returned from fabrication in July 2005 and are currently in evaluation. Another 8031  $\mu$ P, developed by ATK/MRC, was fabricated for a different program in design-hardened 0.18 $\mu$ m ASIC using the same core IP and has been functionally demonstrated. Additional testing of this core, which appears to implement a fully rad-hard 8031 functionality in less than 50 mW, is planned for late August 2005.



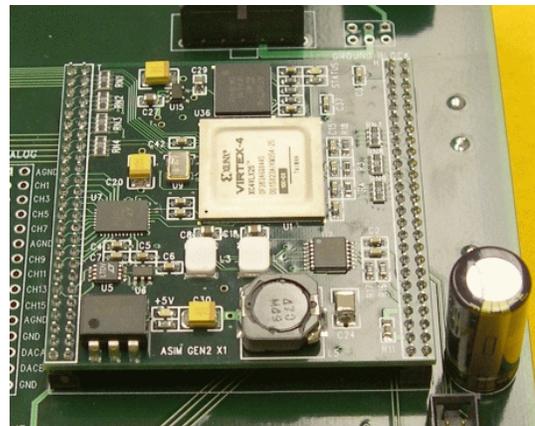
**Figure 11. Design-hardened structured ASIC layout of “Gen 1” SPA-U components**

In parallel with the development of rad-hard core blocks, a special ASIM developmental brassboard is under development based on a Xilinx Virtex IV FPGA (Figure 12). The intent of this design is to provide an IP “staging platform”. The brassboard design contains no separate USB or  $\mu$ P elements. These must be soft-implemented using FPGA gate and memory resources. In this manner, it will be possible to integrate various IP blocks (such as shown in Figure 11) that have been previously fabricated as stand-alone ASICs into a “system-on-a-chip” format. Once the IP functions have been physically verified, the entire collection can be refabricated monolithically to produce a far more compact ASIM as a radiation-hardened design.

The current view of SPA-S implementation is that it will be based upon the combination of a SPA-U interface with a juxtaposed Spacewire link (eight additional wires). It is viewed that in such a configuration, the USB portion of the SPA-S interface can serve as a command and configuration interface, while the SPA-S interface can support up to 625 Mbps data transport (based on successful laboratory brassboards of Spacewire on Virtex II FPGA devices). For SPA-S, the SPA-U power conductors are enhanced to support up to 20A power handling. The rationale for this configuration is that most high data-rate components will likely also have higher power consumption requirements. Routers for SPA-S would combine a high-power handling version of the SPA-U hub in tandem with a non-blocking Spacewire crossbar. An eight-port Spacewire crossbar has been demonstrated in brassboard form (on Xilinx FPGAs), and ATK/MRC is working toward the implementation of stand-alone design-hardened ASICs (in 0.18 $\mu$ m CMOS). The first SPA-S prototypes are planned for demonstration and application insertion as early as 2006. In the meantime, brassboard versions of Spacewire will be co-integrated with SPA-U in a more loosely-coupled configuration in AFRL’s RST laboratory later in 2005.

### Software Activities to Date

Preliminary implementations of the SDM have been developed for Linux-based PC platform. SDM was envisioned to be a distributed software system, under the idea that in future spacecraft, a centralized C&DH processor will not be necessary. Rather, the functions of C&DH can be dynamically amortized across a set of SPA components. To this end, a network of Linux platforms were demonstrated in March 2005 in which different parts of the SDM were run on different processors. A reasonably well-documented version of



**Figure 12. Gen 1 ASIM Developmental Brassboard.**

SDM is available at the time of this writing for independent evaluation.

A number of preliminary versions of xTEDS schema have been developed. By employing XML, the schema is itself extensible, permitting SPA developers to evolve the ontology for a variety of component types. A database with generic “component vocabularies” is planned to permit component designers to structure SPA devices to support a number of expected features. Community agreement on the ontological framework represented by the “set of all xTEDS” is an important objective of the SPA initiative, for without this agreement, most SPA components will be semantically incompatible, complicating the design of applications.

### ***Responsivonics Activities to Date***

Work has progressed in the development of advanced concepts that while not directly related to SPA, have a connection to the problem of reducing time in the construction of avionics systems for spacecraft. The most significant progress made is the demonstration of an adaptive wiring manifold (AWM) based on the integration of over 150 metallic MEMS bistable relays into large printed wiring boards. Two of these boards are shown in Figure 13. The boards feature a number of daughtercards, which implement switchboxes (aggregates of MEMS switches) and connectors to USB and Spacewire interfaces. One of the demonstrations possible with this primitive AWM brassboard is the physical re-routing of all-copper pathways under program control. We have, for example, successfully rerouted full-speed Spacewire connections running streaming DVD video between PC consoles.

Another portion of the responsivonics work involves the development of reconfigurable processors and scalable processors for scalable payload computation tasks. One candidate fusion processor, the Wafer Scale Signal Processor (WSSP)<sup>23</sup>, has been under development by USAF. Using the same structured ASIC process described previously, a version of the WSSP was fabricated as a 3mm x 3mm die, and work is ongoing by AFRL to create a space-qualified version. Interfaced with Spacewire, one of the goals of the SPA project is to make SDM-compatible nodes that can be easily aggregated to form powerful networks. Work is also on-going to build a new generation of the MSP based on Virtex IV FPGA devices.

Another part of the responsivonics work involves an ongoing investigation of interconnections for future generations. We are interested in both extremely high performance interconnect (>> 1Gbps) as well as very lightweight but lower performance approaches. We have pursued, for example, a “SpacewireLite” which is compatible with standard Spacewire but requires far

lower gatecounts to implement. A SpacewireLite core was recently fabricated as a design-hardened test chip.

## **CONCLUSIONS**

In this paper, we have discussed an approach to plug-and-play electronics amenable to implementation in real-time embedded space systems. It is intended that this SPA concept will have broad applicability to all variety of spacecraft bus and payload components, and that it will be possible to assemble and integrate spacecraft far more rapidly than previously believed possible. This radically compressed timescale for spacecraft construction is consistent with the goals of responsive space. This paper moreover focused on how USB is being used as the first SPA approach, though not the only SPA approach possible, as any interconnect system could be established with similar properties. In fact, much of the work in making SPA a reality has little to do with the physical movement of bits across wire (or perhaps in the future, wires will be unnecessary as well). Rather, we have found the more significant challenges in creating an infrastructure conducive to rapid integration. The infrastructure includes the ability to make components self-describing and developing a way for applications to take advantage of new components, dynamically added to a system, without major efforts in rewriting the operational flight program. We defined an approach based on the ideas previously established in IEEE 1451 for embedding datasheets in components, combined with a middleware system for registering these services and then tying applications to these services. These less-visible aspects of SPA may ultimately be the most important. A number of our recent demonstrations are promising in that we have shown that it is possible to at least mimic PnP at a level similar to that in standard use in the PC industry. Our special challenge, as in the case of Sun’s Jini approach, is to do this without custom



**Figure 13. Adaptive wiring manifold demonstration system.**

drivers.

Still, once physical components and labels have been applied, such as USB and Spacewire, it is necessary to find a way to make these components suitable for use in space. We have taken the parallel paths of testing COTS components and designing robust versions of these components (i.e., radhard chips). Leveraging AFRL's substantial base of experience in creating such components, we have made considerable progress on this objective. We look forward to creating the eventual possibility of a rapidly assembled spacecraft, which can be built through a set of plugging actions, made possible through a combination of intelligent modularity, reconfigurability, and standardized (machine-negotiated) interfaces.

## REFERENCES

- <sup>1</sup> "Transformation Trends", presentation by OSD Office of Force Transformation, 17 October 2003, available at the Office of Force website (<http://www.oft.osd.mil>).
- <sup>2</sup> Cebrowski, A.K. and J.W. Raymond, "Operationally Responsive Space: A New Defense Business Model", *PARAMETERS: US Army War College Quarterly*, Summer 2005, Vol. XXXV, No. 2. (<http://carlisle-www.army.mil/usawc/Parameters/05summer/contents.htm>).
- <sup>3</sup> J. Lyke., R. Wojnarowski, and T. Stetcher, "Highly Integrated Packaging and Processing", Government Microcircuits Application Conference (GOMAC) Digest of Papers, March 1999, Monterey, CA.
- <sup>4</sup> Lyke, J. "Reconfigurable Systems: A Generalization of Computational Strategies for Space Systems", *IEEE 2002 Aerospace Conference*, 2002.
- <sup>5</sup> Hilland, D.H.; Phipps, G.S.; Jingle, C.M.; and G. Newton. "Satellite Threat Warning and Attack Reporting", *IEEE Aerospace Conference*, vol. 2, 207-217.
- <sup>6</sup> Cannon, Scott and David Dunn, "Adding Fault-Tolerant Transaction Processing to LINDA", *Software-Practice And Experience* **24**(5).
- <sup>7</sup> J.Lyke, K.Avery, and P. Brezna, "BMDO/AFRL Partnership: Advances in Data Handling Systems for Space Experiment Control", *AIAA Journal of Spacecraft and Rockets*, **39**(4):481-488.
- <sup>8</sup> Abbott, R. "Adaptive computer systems", *IEEE Aerospace Conference Proceedings*, 2002, **4**:1819-1824.
- <sup>9</sup> Bartos, M. "Large Area Network for Online Health Monitoring", AFRL Technical Report *VS-TR-1999-1016*, 1 February 1999.
- <sup>10</sup> Potter, David, "Smart Plug and Play Sensors", *IEEE Instrumentation and Measurement Magazine*, pp. 28-30, March 2002.
- <sup>11</sup> Kastner, Wolfgang and Markus Leupold, "How Dynamic Networks Work: A Short Tutorial on Spontaneous Networks",
- <sup>12</sup> Tanenbaum, A.S. *Computer Networks*, Prentice-Hall, Inc., 3rd edition, 1996
- <sup>13</sup> Ken Arnold, Bryan Osullivan, et.al. *The Jini(TM) Specification*. Addison-Wesley, June 1999. Information on

the Sun Microsystems Jini technology can be found at their website, <http://www.sun.com/jini/index.html>.

- <sup>14</sup> Helal, Sumi. "Standards for Service Discover and Delivery," *Pervasive Computing (IEEE)*, 2002.
- <sup>15</sup> Wilson, Warren, Jim Lyke, and Paul Contino, "MEMS-based Reconfigurable Manifold", presented at MAPLD 2001, Johns Hopkins University- Applied Physics Laboratory, September 11-13, 2001.
- <sup>16</sup> J. Lyke. "Ultra-Minature Instrument Control Processor for Space," Government Microcircuits Application Conference (GOMAC) Digest of Papers, March 1999, Monterey, CA.
- <sup>17</sup> Spacewire – Links, Nodes, Routers, and Networks, European Cooperation for Space Standardization, ECSS Report ECSS-E-50-12A, 24 January 2003.
- <sup>18</sup> Paul LeVan, James Lyke, James R. Waterman, James R. Duffey, and Brandon Paulsen. "The Passive Sensor Subsystem for DITP - Current Status and Projected Performance", 2001 IEEE Aerospace Conference Proceedings (Big Sky, Montana, 10-17 March 2001).
- <sup>19</sup> P. McGuirk, J.C. Lyke, G.W. Donohoe "Malleable Signal Processor: A General-purpose Module for Sensor Integration", Military Applications of Programmable Logic Devices (MAPLD) 2000, Sept. 26-28, 2000.
- <sup>20</sup> Wegner, Ingo. *The Complexity of Boolean Functions*. Wiley, Stuttgart, 1987.
- <sup>21</sup> Shanley, T. *Plug and Play System Architecture (PC System Architecture)*. Addison-Wesley, 1995.
- <sup>22</sup> IEEE 1451 (<http://iee1451.nist.gov>)
- <sup>23</sup> Y. Kinashi, R. Linderman, and J. Lyke, "DITP: A Flexible miniaturized Sensor Fusion Processor for next Generation Interceptor Seekers and Surveillance Sensors", *Proc. Of the 7th Annual AIAA/BMDO Technology Readiness Conference and Exhibit*, Colorado Springs, CO. August 3-6, 1998
- <sup>24</sup> Linderman, R.W. et.al. "A dependable high performance wafer scale architecture for embedded signal processing", *IEEE Transactions on Computers*, January 1998, **47**(1):125-128.