METHODS AND APPROACHES FOR ABSTRACTION OF HARDWARE DEPENDENCIES IN SOFTWARE RADIOS

Victor Giddings
Objective Interface Systems, Herndon, VA
+1 (703) 295-6500
victor.giddings@ois.com

Thomas Kacpura
ASRC Aerospace, NASA Glenn Research Center, Cleveland, OH
+1 (216) 433-6830
thomas.j.kacpura@nasa.gov

Vincent J. Kovarik, Jr.
Harris Corporation, Melbourne, FL
+1 (321) 984-5631
vkovarik@harris.com

Abstract

This paper explores abstraction types and levels within the hybrid processor environment of the software radio in the context of NASA deployment of software radio systems. The premise is that abstraction of hardware is more complex than a layer between the drivers and Board Support Package (BSP) for a specific hardware element and the operating system. Due to the power constraints imposed by space flight, many waveforms implementation are driven towards Field Programmable Gate Array (FPGA) and Digital Signal Processor (DSP) implementations. Consequently, hardware abstraction techniques and components must be applied to these processors as well. Furthermore, abstraction approaches must also be integrated with the safety and reliability requirements associated with human space flight. Recent advances in commercially available software will be discussed. The paper will close with a summary of the current landscape and technology areas that require further research.

1. Introduction

Waveform portability has been one of the central themes driving the development of software radios. The ability to add or change functionality of a radio system after it is deployed is a core capability cited for software radio systems. However, experience has been that porting or development of new capabilities and waveforms for a given radio platform is not as straightforward as originally envisioned. A key element of the portability problem is the fact that, although many software radio developers and integrators assume there is some Hardware Abstraction Layer (HAL) that insulates the functional waveform software from the underlying hardware, the focus of this HAL has been between the software elements on the General Purpose Processor (GPP) and other hardware elements, This paper asserts that the problem lies in the fact that the HAL concept mnust be applied to all processing elements within a typical software radio, especially the signal processing elements, if waveform porting cost is to be reduced.

1.1. Hardware Abstraction Layers

Hardware abstraction layers are not a new concept. As the evolution of operating systems progressed from early custom implementations, common tasks and responsibilities of the operating system (e.g. process scheduling, memory management, I/O control, etc.) migrated to a high-level programming language implementation independent of the underlying hardware. Of course, the operating system still was required to execute on each deployed hardware platform, resulting in the evolution of an intermediate layer that provided a well-known set of operating system interfaces to a range of physical devices with custom drivers that interacted with the hardware.

This early abstraction of hardware dependencies away from the core functions of the operating system was the precursor to the HAL in modern operating systems. One simple but very versatile and useful abstraction was the I/O stream made popular by the UNIX operating system. This simple abstraction provided a simple byte stream interface to reading or writing bytes to an I/O device. This abstraction is applicable to a file, a terminal, a keyboard, and just about any other device.

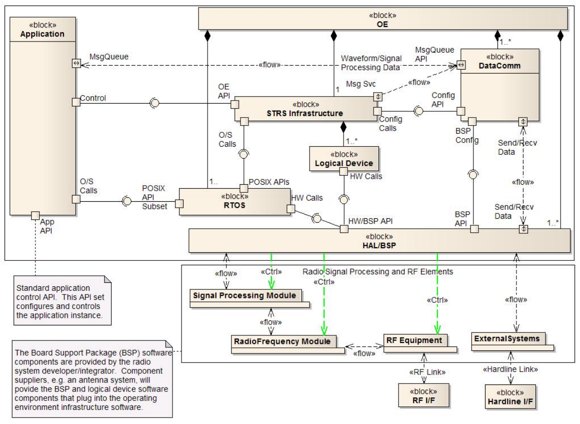

Figure 1. Control and Data Plane Hardware Abstractions Layers

1.2. Software Defined Radio

Although the hardware abstraction layer is a common approach for interfaces to physical devices, there are other applications for this paradigm. One such example is the socket. Using the same stream abstraction, the socket provides a simple interface for sending data to or from another program. The program may be on the same machine or it may be remotely located and accessed over a network.

This simple concept underlies the basic I/O path of data within a software defined radio. As processing components are interconnected, an end-to-end data path is constructed. This is analogous to the socket concept. Thus there are two perspectives of hardware abstraction within an Software Defined Radio (SDR), control plane and data plane. This is illustrated in Figure 1.

There is a subtle distinction between the two abstraction layers, however. In the case of the control plane, the interaction through the abstraction layer tends to be more like an Application Programmer Interface (API) type of interface. That is, a program or application issues a function call to the device abstracted by the HAL. However, in the data plane, the interface tends to behave more like a flow of data between functional components with no dependencies other than propagating the packet of data to the next components for processing when the current component is completed.

The abstraction of hardware becomes more complex as multiple processor types are introduced, e.g. Field Programmable Gate Array (FPGA) and Digital Signal Processor (DSP) processors. The problems of insufficient hardware abstraction was a significant hindrance to waveform porting. The impact of the not abstracting underlying hardware architecture on waveform portability was discussed by Kovarik [1] and lead to the development of the Modem Hardware Abstraction Layer (MHAL) [2] by the Joint Tactical Radio System (JTRS) Joint Program Executive Office (JPEO)

There are a number of constraints that drive the design of an SDR for space deployment. The following section provides and overview of some of the more important constraints and issues with deployment of systems in space.

2. Space Constraints

There are a variety of issues which make the space case unique. An effective architecture must address each of these issues.

Mitigating factors that limit the performance of software defined radios for NASA are the required use of radiation hardened components for in-space use. The transceiver unit is required to effectively operate, for a mission-specific duration, in the radiation environment to meet the mission objectives. This includes elimination or mitigation of long and short term radiation effects (i.e. TID and SEU), performed at the device and/or system level. The cumulative long-term total ionizing radiation dose of the devices can cause permanent degradation or failure. A single event upset can occur when where sufficient energy has been deposited to change the state of a device, such as a bit flip in a memory device. The levels of radiation expected are orbit/mission specific.

The transceiver shall utilize components that are able to survive the temperature extremes of this environment. In addition, the unit shall utilize components and construction techniques that are able to survive the vibration environment subjected during launch.

A major challenge associated with the use of software defined radios is the verification of the software and firmware. While the inherent flexibility allows a radio to be reprogrammed for a new or perhaps unanticipated use, verification of this capability along with some resumption of the nominal operating mode will be required for acceptance. A basic keep alive mode, perhaps in hardware only, may be required if all other functions fail.

Another challenge is to establish an open architecture that manufactures can build to considering the small number of radios required by NASA for the various missions. The architecture must balance using an appropriate level of accepted standards and practices, versus allowing the appropriate customization to meet mission unique requirements inevitable for space missions. Also, the architecture must support long development durations.

2.1. Processors

Today there are a variety of digital signal processing devices, including: application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), digital signal processors (DSPs), and general purpose processors (GPPs). Each processor type has advantages and disadvantages for space transceiver applications. A GPP offers the most flexibility, but runs the slowest. ASICs, custom designed chips for specific processing functions, offer the greatest speed and lowest power requirements, but are limited in predetermined flexibility. FPGAs have features and capabilities that are between the GPPs and ASICs in terms of power consumption, data processing speeds, and reconfiguration capability. The optimal design may use more than one processor type, using the most appropriate component for the various radio stages/functions. As data rates and frequencies increase, more compromises need to be made to balance requirements.

2.2. Size, Weight, and Power (SWaP) Issues

The Space Telecommunications Radio Systems (STRS) [3] architecture shall consider limitations imposed by spacecraft mass, power, volume, and radiation environments. The transceiver hardware size, weight and power requirements are required to conform to the constraints of the host platform. Many NASA space applications have very limited resources with respect to volume, mass, and power. The transceiver must meet the operational objectives within these available spacecraft resources.

The increased power to provide flexibility needs to be weighed carefully against the power available from the spacecraft. This trade affects both the architecture design and the radio implementation for a specific mission.

3. Space SDR Architectures

This section investigates potential architectures for space-base software radios. As might be expected, the severe weight and power constraints imposed by space deployment are significant drivers in the hardware and software elements of an integrated SDR.

The STRS, shown in figure 2, is an open architecture for NASA space based radios . This architecture provides a common, consistent radio framework to develop, qualify, operate and maintain complex reconfigurable and reprogrammable radio systems. This architecture standard provides a detailed description and set of rules for the architecture, focusing on describing the key architecture components and subsystems through their functionality and interfaces for both the hardware and the software architecture including waveform applications.

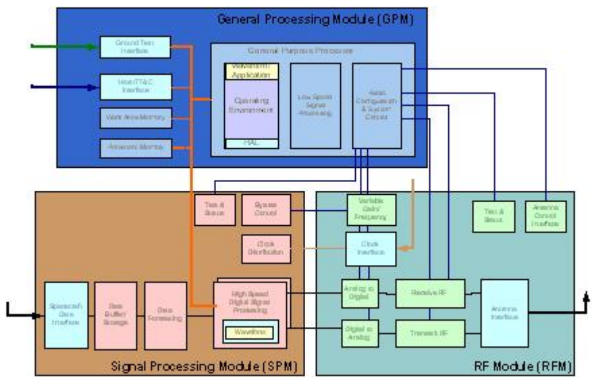

Figure 2. STRS Software Radio Architecture

The architecture must accommodate a range of reconfigurable processing technologies. The waveform developer has the option of allocating the waveform application to execute in any of the various processors, or even distributing portions of the application in several of the processors. The waveform developer can make this allocation choice depending upon the waveform requirements and the capabilities of the platform where this application is run.

3.1. Hardware

Key trades of using software defined radio for space missions need to be weighed against the some of the negative issues of reconfigurable radios. Changes in operation need to be carefully considered and executed so that the operation is not compromised. If the radio is changed to an unrecoverable state, then command and control of a space platform would be eliminated, and the spacecraft could be considered lost. The resources on a spacecraft are usually very limited and tightly controlled, and if extra power is required for a reconfigurable radio compared to a standard radio, and not all the features used during operation, then some of the advantages will be negated.

Another issue is the reconfigurability versus power consumption trade. The more flexibility radios have a higher reliance on general purpose processors, and these use more power than the radio designed with ASIC or FPGAs, which are less flexible. Use of these components requires that flexibility be a design parameter and considered as the radio is designed. This is compounded by the requirement to use electronic components suitable for the space environment, which is generally significantly slower than state-of-the-art commercial off the shelf (COTS) hardware used for terrestrial radios.

Currently reconfigurable signal processing is primarily performed in specialized signal processing hardware for the frequencies and data rates used in NASA space missions, and this is expected to continue for some time. In addition to providing capability, specialized signal processing is generally more power efficient than general purpose processing. Likewise, the use of FPGA-based specialized signal processing is generally more power efficient than Digital Signal Processor (DSP) based signal processing. The use of these devices support more power efficient radio implementations, but make the FPGA firmware and DSP code are tightly coupled to hardware and very difficult to port.

3.2. Infrastructure

The software infrastructure of a space SDR must provide comprehensive management of the physical and logical resources of the radio system while minimizing operating overhead and memory footprint. Current SDR infrastructures such as the Software Communications Architecture (SCA) [4] and the Object Management Group (OMG) Software Radio Platform Independent Model (PIM) specification [5] support these goals, but current implementations are much too large to meet the overhead and memory footprint constraints.

The STRS, developed by National Air and Space Administration (NASA), describes a software radio infrastructure approach that considers the limitations imposed by space deployment. Within the STRS specification, the abstraction layer is addressed within the context of the GPP environment and identified as a component within the signal processing element of the radio. However, this latter reference to a HAL is not as clearly delineated as within the GPP.

A major high-level STRS Architecture objective is to enable waveform application portability. These waveform applications are envisioned to use specialized signal processing devices such as FPGAs for executing the waveform applications. FPGAs contain components and capabilities to manipulate and manage digital signals that have higher processing capabilities and lower power consumption than general purpose processors.

3.3. Waveforms

As noted by Selby [6], reuse of software components incurs some cost. In the context of a software radio, the complexity and corresponding reuse cost increases significantly. This is due, in large part, to the impacts due to differences in the underlying hardware architecture and the tight coupling between the waveform implementation and the physical architecture, particularly in the case of FPGA processors. In order to promote easier portability, the waveform implementation must be abstracted away from underlying hardware.

Key to enabling this abstraction is a well-defined hardware abstraction layer within the digital signal processors. The HAL within a FPGA will have a different perspective. Since the FPGA implements a synchronous, parallel, data flow machine, the HAL takes on less of an API perspective and more of a processing node in a computational chain.

4. Data Transport Approaches

The data transport infrastructure engineering tradeoffs for the data plane are different from the trade-offs for control plane. The main concerns for the data place are 1.) for the incoming data, to ensure that the data is demodulated from the radio signal and transferred to data output(s) of the radio reliably, and 2) for the outgoing data, to ensure that the data is properly transferred from the digital input(s) and modulated onto the radio signal. In doing so, the infrastructure must meet the bandwidth requirements of their respective channels while minimizing cost of the solution; minimizing latency is usually a secondary concern. For the control plane, bandwidth is usually a secondary concern; latency, predictability of latency, and reliability of delivery are the primary concerns.

4.1. Protocol v. Transport

There are also different layers within the data transport infrastructure. An electrical signaling layer is concerned with indicating the transfer of individual bits between components and are implemented in a number of ways, e.g., by a bus or direct copper connection between components. Overlying the raw signaling, there is a link protocol layer that ensures reliability of the transmission and that the flow of data from the source does not overrun the capability of the destination. Finally, there is what is usually termed the "middleware" layer that mediates data transfer between the radio application components. The middleware layer further mediates the impedance mismatch between the representations of data by the radio application components and the protocol layer used to transfer the data between operational components. Most middleware deals with structured data, and so serialization and deserialization (serdes) are a major concern along with reliable and timely delivery.

The engineering tradeoffs at the different layers differ. Implementation of the signaling layer is driven by cost and availability of the physical infrastructure required for the transport. The concerns at the link protocol layer tradeoff complexity for reliability, since complexity drives the cost of the required hardware infrastructure to support the protocol. For example, the number of gates required to implement a link protocol on an FPGA will drive the selection of a connection to the FPGA. The tradeoffs at the middleware layer are similar: complexity, and its attendant cost in implementation resources, is traded against robustness. However, the ease of use, i.e., the ability of a developer to integrate a component with a middleware technology, along with the longevity and availability of middleware implementations, affects the usefulness of the middleware technology.

4.2. Implementations

The processor class also affects the engineering tradeoffs for the different layers of data transports. Different GPPs support a wide variety of data link transport protocols and signaling interconnects. Significant bandwidth communication onto or off of a GPP is usually achieved only with dedicated peripheral devices that have direct access to the memory of the processor. The same is true of DSPs. FPGAs offer a large number of pinouts that can be connected to I/O peripherals, so the availability of a signaling and data link transport is largely a matter of the availability of "driver" logic blocks for the peripheral hardware. To complement the throughput capabilities of FPGAs, dedicated GPP I/O transports, such as HyperTransport, have been developed. The middleware situation, up until recently, was not as uniform. High-performance small footprint Common Object Request Broker Architecture (CORBA) implementations have a proven track record on GPPs. The CORBA products have been targeted DSPs resulting in reports of substantial speed improvements while maintaining small footprints. The non-"vonNeumann architecture" of FPGAs complicated the application of middleware principals to FPGAs. However, recently CORBA-based products have been released that are targeted to the FPGAs used in digital signaling applications. These products provide a level of interoperability with other middleware products that allow a degree of "technology transparency". This transparency allows systems to be initially developed or prototyped as software-only components. Performance-critical components can be migrated to higher-capability technologies, DSPs or FPGAs, over the course of product or product line life cycle.

4.3. Reuse and Abstraction

The software infrastructure of a space SDR must provide consistent data transports for both data plane and control plane traffic at multiple levels. This consistency provides the transparency that allows signal processing components and other types of components to be targeted to the hardware appropriate to the performance requirements while meeting cost, SWaP, and space environment constraints for the particular SDR application. Further this transparency preserves investments in component implementations in the face of requirements changes and technology advances. Current SDR infrastructures provide this consistency, but only for the GPP-based components.

The HAL concept needs to raised in abstraction level to accommodate not only consistency at the data signaling and link transport level, but also accommodation of software component interoperability regardless of hosting of the software component on a GPP, a DSP, or FPGA. Standardization of a middleware technology such as CORBA would seem feasible and would provide a consistent component interconnection view across all components regardless of the implementation technology.

5. Summary

We have presented architectural approaches and tradeoffs for the development of a space-qualified SDRSDR. In conjunction with the constraints imposed by space deployment, waveform applications require power efficient processing. In order to have portability of these applications, the abstraction of specialiized signal processing devices must be addressed. There are two key perspectives to the HAL concept, control and data, and that there is a fundamental difference in the types of interactions performed through each of these interface planes.

6. Acronyms

API	Application Programmer Interface
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
HAL	Hardware Abstraction Layer

JTRS	Joint Tactical Radio System
JPEO	Joint Program Executive Office
MHAL	Modem Hardware Abstraction Layer
NASA	National Aeronautics and Space Administration
OMG	Object Management Group
PIM	Platform Independent Model
SCA	Software Communications Architecture
SDR	Software Defined Radio
STRS	Space Telecommunications Radio Systems

References

[1] V. Kovarik. The impact of hardware architecture on waveform portability. In *Software Defined Radio Forum Technical Conference*, Orlando, FL, November 2006.

[2] Joint Program Executive Office (JPEO), Joint Tactical Radio System (JTRS), Space and Naval Warfare Systems Center, San Diego, CA. *Joint Tactical Radio System (JTRS) Standard Modem Hardware Abstraction Layer Application Program Interface (API)*, version 2.11.1 edition, May 2007.

[3] NASA, Glenn Research Center, Cleveland, OH. *Space Telecommunications Radio System (STRS) Architecture*, version 1.01 edition, December 2007.

[4] Joint Program Executive Office (JPEO), Joint Tactical Radio System (JTRS), Space and Naval Warfare Systems Center, San Diego, CA. *Software Communications Architecture Specification*, version 2.2.2 edition, May 2006.

[5] Object Management Group. *PIM and PSM for Software Radio Components Specification*, version 1.0 edition, 2007.

[6] R. Selby. Empirically analyzing software reuse in a production environment. In W. Tracz, editor, *Software Reuse: Emerging Technology*, pages 176–189. IEEE Computer Society Press, 1988.