Hardware Abstraction Layer Working Group Report on Results of Request For Information

SDRF-04-P-0009-V1.0.0

(Formerly SDRF-04-A-0009-V0.00)

4 October 2004

SDR Forum
Technical Committee
Hardware Abstraction Layer Working Group

Report on Results of Request For Information

July 17, 2004

Contact: John Chapin, jchapin@vanu.com, +1 617-864-1711 x202.

Abstract

The HALWG RFI sought information on tools and technologies that assist with the portability of high speed signal processing code (FPGA, DSP, or other) in the signal processing subsystem (SPS). The RFI also sought information for improving the portability of higher layer code (HLPS) that interacts with the SPS. This document consists of

- analysis of the RFI responses
- recommendations for further action by the SDRF related to SPS code portability
- a 1 page summary of each of the RFI responses in a standard format.

Acknowlegements

Significant contributions to the preparation of this document were provided by: Christian Serra, Kevin Maier, Mark Littlefield, David Fitkin, Bruce Fette and Eric Nicollet, and the other members of the Hardware Abstraction Layer Working Group.

Table of contents

SDRF HALWG RFI Report At-A-Glance

The taxonomy section of the report defines the meanings of the check-box columns.

		Page	Category				Maturity			SCA relation		
			Autogeneration	Modeling	Environment	Control/Other	Research	Product	Standard	Close	Compatible	Conflict
Berkeley Wireless Research Ctr.	A design and runtime environment for software radios on reconfigurable hardware	14	●				●				●	
Xilinx, Inc.	Tools and design methodologies for specifying and implementing the SPS.	15	●					●			●	
BAE Systems - CNIR	Waveform Description Language (2)	16	●				●					●
The MathWorks, Inc.	HDL generation for FPGAs from Simulink block diagram models	17	●					●			●	
The MathWorks, Inc.	Autogeneration of DSP and GPP code from Simulink	18	●					●			●	
Siemens AG	Development system for hardware independent signal processing algorithm description	19	●				●					●
A3S Consortium	UML model of Hardware Abstraction Layer and associated tools	21		●			●				●	
QinetiQ	Waveform Description Language (1)	22		●			●					●
Infineon Technologies AG	Signal Processing System Description	23		●			●					●
The MathWorks, Inc.	Simulink framework for sharing specifications and requirements	24		●				●			●	
The MathWorks, Inc.	Simulink for hardware-independent end-to-end system models	25		●				●			●	
The MathWorks, Inc.	Data types and APIs for fixed-point numerics, simulation, and code generation	26		●				●			●	
The MathWorks, Inc.	Open platform for verification, validation and test.	27		●				●			●	
Mercury Computer Systems, Inc.	Portable Components using Containers for Heterogeneous Platforms	29			●		●				●	
Mercury Computer Systems, Inc.	Digital IF Specification	30			●		●				●	
Object Management Group	OMG SWRadio specification	31			●				●	●		
General Dynamics	Hardware Abstraction Layer Definition Within SCA	32			●		●			●		
Celoxica Ltd.	Platform Abstraction Layer	33			●			●			●	
Spectrum Signal Processing	Hardware Abstraction Layer Software Development Library	34			●			●			●	
The MathWorks, Inc.	API for component library supporting both simulation and run-time execution	35			●			●			●	
Thales Communications S.A.	Definition of Signal Processing Subsystem (SPS) Hardware Abstraction Layer	36			●	●	●			●		
E2R Consortium	High level abstraction of signal processing elements	38				●	●					●
Harris Corporation	A method of specifying FPGA control interfaces using XML	39				●	●			●		
Space Coast Comm. Systems, Inc.	Device-centric SDR solutions and the Software Communications Architecture	40				●	●			●		

Index by Organization

Analysis and recommendations

Overview

The SDRF HAL Request For Information sought information on tools, technologies and standards that assist with the portability of high speed signal processing code (FPGA, DSP, or other) in the signal processing subsystem (SPS). The RFI also sought information for improving the portability of higher layer code that interacts with the SPS. The techniques that assist in portability in these areas are collectively known as the Hardware Abstraction Layer for software defined radio.

The RFI was distributed to radio manufacturers, software developers, technology providers, researchers, and other interested organizations. It drew enthusiastic response from the community. 24 responses were received from 17 organizations. Among the submitters were a university, two research consortia, a standards body, tool vendors, system integrators, and technology providers. This shows wide agreement that the hardware abstraction layer is an important area of concern for software radio.

Related efforts

The System Interface Working Group of the SDR Forum (SDRF SIWG) is studying the broad set of interfaces to all radio components. In addition, activity related to the radio signal processing subsystem is occurring in two other organizations.

- The Software Radio Domain Special Interest Group of the Object Management Group (OMG SWR DSIG), an international standards body, is carrying out a standardization process on all aspects of the software for software defined radios.
- The Joint Tactical Radio System Joint Program Office of the US DOD has developed a proposed set of standard interfaces and design techniques for military radios as part of its Software Communications Architecture (SCA) version 2.2.

The SDRF has established liaisons with these organizations to promote the results of this RFI and potential future HAL standardization efforts.

Areas Covered by RFI

Figure 1 shows an architecture model of a software defined radio. It shows the OSI layer 1 (modem) function of the signal processing subsystem (SPS). The SPS may additionally perform application-level audio or video processing and similar functions, not shown in the figure. The SPS works together with the Higher Layer Processing Subsystem (HLPS). The HLPS performs work such as protocols, routing and control. The HLPS typically executes on standard general-purpose processors or microcontrollers, in contrast to the DSPs, FPGAs or other processing technology in the SPS.

Figure 1 also shows the areas covered by the RFI. It sought information regarding existing and proposed interfaces, tools, and techniques that may facilitate software portability across platforms having varied signal processing subsystems. Responses were requested in two independent areas:

(a) the portability of HLPS code with respect to SPS variability, and
(b) the portability of the SPS code itself.

Figure 2 shows a software model of a software defined radio and indicates some potential places where APIs and tools could be applied to improve software portability.

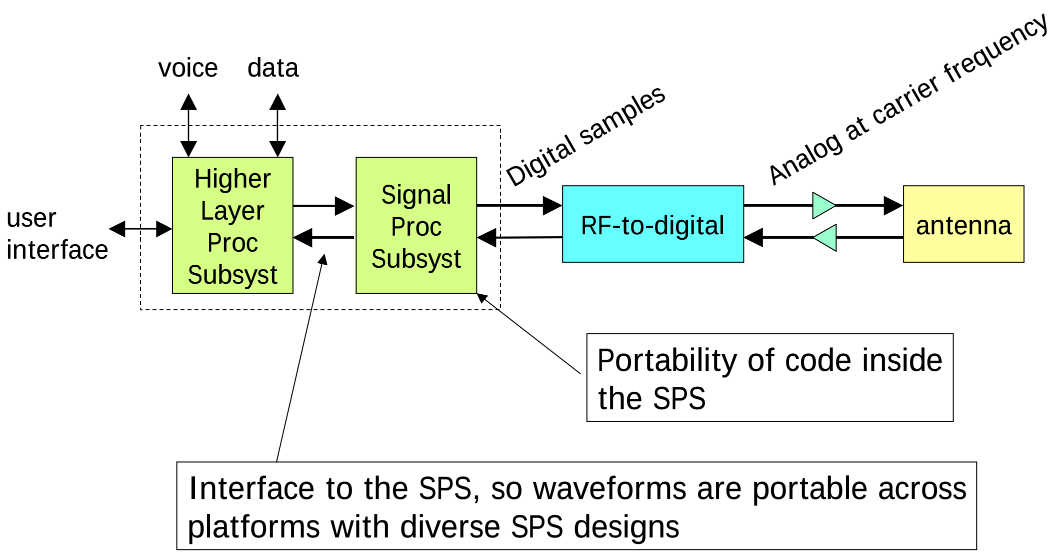
voice
data
user interface
Higher Layer Proc Subsyst
Signal Proc Subsyst
Digital samples
RF-to-digital
Analog at carrier frequency
antenna
Portability of code inside the SPS
Interface to the SPS, so waveforms are portable across platforms with diverse SPS designs

Figure 1. SDR Architecture Model and Areas of RFI Interest

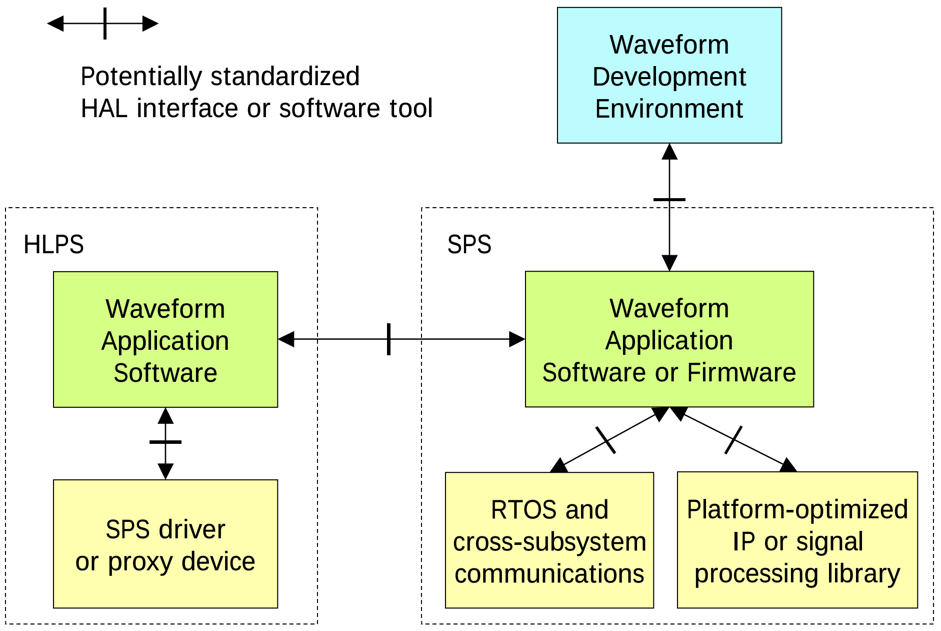
Potentially standardized HAL interface or software tool
Waveform Development Environment
HLPS
Waveform Application Software
SPS driver or proxy device
SPS
Waveform Application Software or Firmware
RTOS and cross-subsystem communications
Platform-optimized IP or signal processing library

Figure 2. SDR Software Model and Potential Roles for a SPS HAL

Standardization

The RFI asked whether the HAL should be standardized. A strong consensus emerged in favor of standardization, with support coming even from vendors of currently successful but non-standardized products. However there was some disagreement about the appropriate standards body. Many responders recommended the OMG but a minority favored IEEE or ETSI due to interactions with hardware design issues.

Areas of response

The responses describe a diverse set of ongoing, completed, or proposed efforts. There are many sources of porting cost and therefore many different ways to reduce that cost. The responses fell largely into four categories.

- Autogeneration 6 responses
  These responses describe methods for automatically generating signal processing implementations from models and/or high level source code. They reduce porting cost by automating some of the work of writing the implementation for the target processor.
- Modeling 7 responses
  These responses describe ways of exploiting abstract models other than by code generation tools. They reduce porting cost by reducing design effort or risk. Models are used particularly to reduce costs caused by real-time performance requirements and by specification or design ambiguity.
- Environment 8 responses
  These responses propose standard features for the operating environment of software components inside the SPS. They reduce porting cost by reducing the changes required when porting a software component across different platforms.
- Control 2 responses
  These responses describe methods for configuring and controlling the SPS in an abstract fashion. They reduce porting cost by reducing the changes required in the rest of the system when the SPS changes.

Autogeneration and Modeling

Some responses described mature commercial products for autogeneration. We are aware of other products in this area from companies that did not respond to the RFI.

For modeling, we received submissions related to a wide range of design stages and development needs in which models may be used to reduce porting costs. Some of the more promising efforts are at an early research stage and will only mature over time.

It is clear that different responders envisioned different design flows for waveform developers and system integrators. By “design flow” we mean the sequence of decisions made and artifacts created during engineering or porting, along with a description of the features and capabilities of the tools used to assist at each stage. An example top-down design flow might start with a problem model, then develop models of specific components, algorithms enabling those components, map algorithms to processing devices, and then autogenerate code. There are also bottom-up design flows which start from a platform performance model, intended to aid in porting.

We believe there are multiple design flows corresponding to different developer requirements or usage scenarios driven by different types of SDR devices and user requirements. We believe that a better understanding of design flows would be valuable for the SDR community. A categorization of design flows along with a description of the artifacts developed and tools used in each one would provide:

- Increased clarity when discussing autogeneration and modeling approaches or analyzing tool interactions
- A framework for developing SDR industry specific tool requirements to drive vendor development efforts
- A roadmap for vendors to plan improvements that reduce porting cost

We recommend that the SDRF form a study group to investigate design flows for SDR.

Environment

There were two main types of responses in the environment category. There are mechanisms and APIs operating at a low level. These are represented by the Spectrum and Celoxica connectivity products and the Mercury proposal to standardize the container for FPGA components. Separately, there are mechanisms and APIs oriented to a high level which standardize the services used by the waveform and perhaps the interfaces to some of the waveform components themselves. There are three major proposals among the RFI responses, from Thales, GDDS, and OMG.

The terms “low-level” and “high-level” used here are not well defined. There is no clear understanding of the number of abstraction levels or the categorization of functions into levels. We recommend that the HALWG work to establish a reference model and/or layer definitions to clarify this area which will serve as a basis for future work. For now, we use these terms to refer to the two groups of responses just delineated.

The low level mechanisms offer the potential for concrete, near-term reductions in portability costs if standardized across multiple platforms. Noting that the JTRS JPO SCA 3.0 extension process is active in this area, we recommend that the SDR forum wait for the results of that effort before taking further action.

A high level standard will be necessary for true portability of waveform software component implementations. However, there is clearly significant work to be done in this area before the current proposals are mature enough to be considered as candidate standards. There are significant design and efficiency questions regarding potential approaches.

We recommend that the SDR Forum continue discussions regarding potential high-level HAL standards with the organizations developing them as the proposals are explored and matured. It would be appropriate for the HALWG to take the lead in this effort, for example by inviting presenters to future SDRF general meetings and reviewing their publications. We further recommend that the SDR Forum Technical Committee consider sponsoring a reference implementation in this area at some point in the future to experiment with a promising potential approach or approaches.

Control

Both RFI responses we received described early-stage efforts. However, we note that the E2R WP4 effort, as a European-wide research project, is explicitly aimed at providing input to future European standardization efforts. At present the E2R effort is not tightly coupled to the OMG SWRADIO submission or to the SCA effort.

We recommend that the SDR Forum work to facilitate interaction between OMG, SCA development efforts, SDRF members active in the area of SPS control, and the E2R WP4 (and to some extent WP2). The goal is to promote harmonization and reduce duplication of effort.

Summary of Recommendations

- The SDRF should form a study group on design flows and tools. This will increase clarity of discussions, provide a framework for developing SDR industry-specific requirements, and provide a roadmap for vendor tool improvements.
- The HALWG should develop a consensus HAL reference model and/or layer definitions to clarify the categorization of the various functions performed by a HAL.
- The SDRF should wait for the results of the JTRS JPO SCA 3.0 extension effort before pursuing further investigation of low-level HAL functions.
- The HALWG should interact further with organizations proposing high level HAL standards and assure their efforts are presented to SDRF members for consideration.
- The SDRF technical committee should consider sponsoring a reference implementation of a high-level HAL at some point in the future.
- The SDRF should exploit its close ties to OMG, JTRS, and its new partnership with E2R to promote harmonization and reduce duplication of effort regarding configuration and control of signal processing subsystems.

Taxonomy

The next sections of this report give a one-page summary for each RFI response. Each response is categorized in three ways: architectural category, maturity, and relationship to the JTRS SCA.

These category values were assigned by the HALWG during the review process. The values should not be interpreted as statements by the authors of the RFI responses.

The architectural category indicates which aspect of the system is addressed by the tool or technology. The categories are:

Autogeneration	method for generating implementations from models and/or source code
Modeling	method for exploiting models (other than autogeneration)
Environment	interface between SPS processing components and things they talk to
Control	deployment/configuration/control of SPS software/hardware by HLPS
Other	

The maturity category indicates the stage of development of the tool or technique.

Research
Early product
Widely used product
Standard in development
Standardized

The relationship to the SCA indicates how the tool or technique fits with the emerging SCA standard.

Close	specifically designed to extend/support/interface with the SCA
Compatible	can be used in conjunction with the SCA
Conflict	significant changes required to the SCA or to this if they are used together

Autogeneration

These responses describe methods for automatically generating signal processing implementations from models and/or high level source code. They reduce porting cost by automating some of the work of writing the implementation for the target processor.

A design and runtime environment for software radios on reconfigurable hardware

Authors: Kevin Camera, Hayden So, Professor Bob Brodersen

Organization: Berkeley Wireless Research Center,
Department of Electrical Engineering, U. C. Berkeley

Architectural category: Autogeneration

Maturity: Research

SCA relationship: Compatible

What is described?

BWRC is developing a language, compiler, and operating-system like runtime services to ease the process of developing FPGA implementations of wireless systems. The goal is to provide high level programming features – such as function calling, debugging (real-time and single-step), file system access, and device abstraction – while maintaining direct and efficient mapping to hardware structures. BWRC does not distinguish between the Signal Processing Subsystem and higher layer function. The complete wireless system is implemented in the FPGA or other reconfigurable hardware.

How would it improve code portability if adopted widely?

Code portability is improved since there is no longer any variation between different processor models or instruction set architectures. Plus, by implementing the HLPS and SPS on the same platform, uncertainty in the interface (such as buffering requirements or timing mismatches) is minimized. The performance of portable code is also improved, since there is no need for software wrappers or translation code when processing elements are directly mapped onto the FPGA as needed.

How does it compare to widely used alternative approaches?

Direct-mapping processing elements on FPGAs has been shown to be more energy efficient than implementing algorithms on sequential processor architectures, due to the exploitation of spatial resources and parallel processing. Instruction-based processors are also notoriously difficult to program under real-time constraints, with caches and interrupts preventing deterministic execution time in most cases. The primary challenge in using FPGAs as the processing fabric for higher layer processing is in the design environment and programming model. This approach eases those challenges.

What is the level of technology maturity, standardization, commercial acceptance?

The prototyping engine (Berkeley Emulation Engine, or BEE) has been in use at BWRC for 4 years. The design flow used with this prototyping platform has been used by in-house users, graduate and undergraduate courses, and external academic and industrial partners. A much more modular and scalable sequel to BEE is currently in the works for supercomputer-class performance and more general computing applications. The tool flow itself, which includes the language definition and compiler implementation, is still being specified and is not yet developed. The core kernel of the operating system has entered development, but the interfaces to user processes and external device drivers are yet to be created.

Other issues to consider:

None.

Tools and design methodologies for specifying and implementing the SPS.

Authors: Dr. Jim Hwang, Dr. Chris Dick

Organization: Xilinx, Inc.

Architectural category: Autogeneration

Maturity: Widely Used Product

SCA relationship: Compatible

What is described?

A tool, System Generator, which is used in conjunction with the MATLAB/Simulink environment to provide high-level simulation libraries, modeling and co-simulation interfaces, and code generation technologies for Xilinx FPGAs. The System Generator tool allows high level design of FPGA blocks (platform-based or model-based design), along with efficient automatic code generation for FPGA designs. Xilinx also proposes that standard low latency hardware interfaces be used to provide the distributed flow control necessary for highly modular and reconfigurable signal processing data paths.

How would it improve code portability if adopted widely?

Some blocks in the System Generator library are pure synthesizable RTL, and standardizing both the data transport and control as part of the SPS library specification would greatly assist in interoperable realizations, hence lead to more portable code. With respect to performance, utilizing high level algorithmic design in the System Generator tool, along with high-performance hand crafted FPGA blocks allows for rapid development of very efficient FPGA cores.

How does it compare to widely used alternative approaches?

Prevailing FPGA design methodologies involve a decoupling of system modeling and implementation. As a result, the development cycle is not optimal. Platform based design using System Generator allows an implementation to be largely developed at the same time as modeling, allowing (by a LANL estimate) a 2-4x productivity improvement over traditional FPGA design methods. Xilinx also believes that using the industry standard Simulink framework allows better control over many aspects of design that lead to more efficient hardware implementations.

What is the level of technology maturity, standardization, commercial acceptance?

System Generator has been commercially available since the year 2000 and has undergone 8 major revisions. It is now used by hundreds of companies worldwide.

Other issues to consider: Proprietary content, etc.

System Generator is proprietary technology of Xilinx, Inc. It is currently possible to build user-defined functions that are incorporated into System Generator via a Black Box interface, and based on requests from their customer base, Xilinx is exploring incorporating APIs for user defined functions.

Waveform Description Language (2)

Authors: Robert Prill

Organization: BAE Systems - CNIR.

Architectural category: Autogeneration

Maturity: Research

SCA relationship: Conflict

What is described?

BAE describes a component-based application framework specifically for waveforms that encompasses not just the SPS or the SPS/device interfaces, but the entire physical layer of a radio system with a simple API interface to the data link and network layers. They establish:

- A Canonical Radio Template incorporating components that are familiar to radio engineers
- A means of interconnecting components and orchestrating them to simulate the waveform
- An approach to automatically generate radio code from behavioral simulation code

How would it improve code portability if adopted widely?

The WDL approach to portability is to design basic components and utilize a common component-based framework. MATLAB, Simulink and a compiler called AccelFPGA from AccelChip are used to autogenerate target-specific implementations from high level behavioral simulations. The WDL approach includes a web-based component library system aimed at reducing engineering effort.

How does it compare to widely used alternative approaches?

WDL has signficant overlaps with the SCA since it covers the entire physical layer. WDL differs from the SCA in mechanisms for defining physical layer components, interconnection and timing control. Performance maximization and latency minimization are the rationale for the differences. The WDL approach goes beyond the SCA through applying to both conventional Von Neumann processors (GPP and DSP) and to parallel architectures like FPGAs.

What is the level of technology maturity, standardization, commercial acceptance?

The Waveform Description Language is an experimental platform and has had only lab exposure to-date.

Other issues to consider:

This submission is an alternative component-based framework for the entire radio physical layer. However, the WDL approaches to abstracting the SPS/device interfaces and data transfer between FPGAs, processors, and other components could be useful for a standardized HAL even if the entire WDL approach is not adopted.

HDL generation for FPGAs from Simulink block diagram models

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Autogeneration

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Two MathWorks products: the existing Simulink system and an upcoming product called Filter Design HDL Coder. These work together with third party software specific to each target platform.

These products enable users to automatically generate FPGA implementations from Simulink models. Users create a hardware-independent design and substitute in platform-specific optimized models that generate the appropriate HDL and IP core representation. The upcoming HDL Coder product goes further by automatically generating synthesizable VHDL or Verilog from fixed point filters designed using the MathWorks Filter Design Toolbox.

How would it improve code portability if adopted widely?

The Simulink approach facilitates design portability rather than portability of the implementations of individual components. It is particularly useful in situations where the partitioning between FPGAs and software-programmable processors may change in the future. The HDL Coder will improve portability of implementations of certain component types.

How does it compare to widely used alternative approaches?

The alternative approach is manual coding of the full design in HDL. By switching to automatic generation based on Simulink:

- More engineers can do the work since few engineers have the requisite combination of DSP and HDL knowledge to implement these designs directly in an HDL.
- Manual coding of HDL implementations makes it difficult to gain insight into the system behavior for design optimization or code porting.
- Most SDR systems involve both FPGAs and a conventional DSP or microprocessor, which cannot be represented in an HDL alone. Using Simulink unifies the design for the different processor engines.

What is the level of technology maturity, standardization, commercial acceptance?

Simulink is the de facto standard for high level designs of FPGAs. Both Xilinx and Altera, the leading FPGA vendors, have adopted this approach and have offered products for 3 years based on Simulink. These are in wide commercial use. The Filter Design HDL Coder is in beta test, and will be released in June 2004.

Other issues to consider: Proprietary content, etc.

Although these are products from one vendor with proprietary content, they are used as a de facto standard throughout the industry. The MathWorks has indicated potential willingness to create an open file format which captures all the information in a Simulink design. If this file format were supported with import/export into the Simulink tools it would mitigate proprietary concerns.

Autogeneration of DSP and GPP code from Simulink

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Autogeneration

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Using The MathWorks Real-Time Workshop Embedded Coder product, users can automatically generate software from Simulink models. The default code is ANSI/ISO C. Using the MathWorks open interfaces, customers can substitute target-optimized code for the portable code by creating a library of optimized components to be used at code generation time.

The real-time code can be executed directly on target hardware for verification and debugging, and can easily be integrated with external software. Optional products provide target-specific support for TI DSPs and Motorola, Hitachi, and Infineon microcontrollers.

How would it improve code portability if adopted widely?

This approach facilitates hardware independence and design portability, not merely code portability. From the same simulink model, it is possible to autogenerate both optimized C code for a variety of targets and VHDL for FPGA implementations. This reduces the cost of porting the software to a new platform with a different partitioning of functions across GPPs, DSPs, and FPGAs.

How does it compare to widely used alternative approaches?

The primary alternative is hand-coded C. In most cases the output of the Embedded Coder has performance comparable to portable hand-coded C. Where required, hardware-specific optimizations can be added by the user.

The primary advantage is an increase in engineering productivity. This approach automates the creation and testing of embedded software applications. This gives system and algorithm engineers access to embedded hardware without extensive coding, and allows software engineers to focus on system integration and optimization.

Use of automatically generated code has significant cost advantages, because it results in order-of-magnitude reductions in development time and minimizes errors introduced by manual translation of specifications into embedded DSP or control code.

What is the level of technology maturity, standardization, commercial acceptance?

The Real-Time Workshop code generation technology has been available and continuously improved over the last ten years. The optimizations for production use were introduced in 2000 and target specific optimizations for DSPs and microcontrollers were introduced in 2001.

Other issues to consider:

MATLAB, Simulink, blocksets and related products are proprietary technology of The MathWorks. These products and features cannot be implemented or utilized without a license from The MathWorks.

Development system for hardware independent signal processing algorithm description

Authors: Dr. Matthias Weßeling

Organization: Siemens AG, ICM MP PD TI2

Architectural category: Autogeneration

Maturity: Research

SCA relationship: Conflict

What is described?

This submission explores the general concepts of a Virtual Radio Engine (VRE) to provide automatic code generation for specific parallel hardware processing resources.

VRE concepts are based on the separation of all hardware specific and all algorithm specific information and require the description of the application and the hardware processing resources capabilities using a Virtual Radio Engine Language (VREL).

Description of the application should contain only information about algorithms to use, and the interfaces it requires, without assumptions of the hardware platform it should run on.

Description of the hardware platform should address optimized libraries, synchronization and parallelization features, code generation options,…

Starting from these descriptions, VRE code generation should perform the following tasks:

- Analyze latency and Performance Timings
- Generate Parallelization
- Restructure Algorithms
- Determine Schedule, Buffer Sizes, Synchronization Mechanisms
- Map processing functionalities to the hardware architecture
- Generate Processor Specific Implementation Code.

How would it improve code portability if adopted widely?

Inside the VRE concepts, hardware features should be considered in detail with the code generation process.

How does it compare to widely used alternative approaches?

Current code generation tools focus on local optimizations of blocks, but do not consider the overall system description and timing requirements.

What is the level of technology maturity, standardization, commercial acceptance?

There is already an abstracted description of the WLAN 802.11a signal processing algorithms with a protocol stack up to the MAC layer available at Siemens AG.

An existing proof of concept tool chain generates code for different parallel hardware architectures. This to0l is still in a development status and not all aspects are implemented yet.

Other issues to consider:

As the VRE concept does not exploit run time APIs to isolate independently developed components, VRE conflicts with the object-oriented SCA approach.

Modeling

These responses describe ways of exploiting abstract models other than by code generation tools. They reduce porting cost by reducing design effort or risk. Models are used particularly to reduce costs caused by real-time performance requirements and by specification or design ambiguity.

UML model of Hardware Abstraction Layer and associated tools

Authors: Antoine Delautre, Jean-Etienne Goubard, Christophe Moy, Jean Luc Philippe, Guy Gogniat, Nicolas Bulteau

Organization: A3S
(a consortium of Thales Communications SA, Université de Bretagne Sud Lester, SOFTEAM, and Mitsubishi Electric ITE)

Architectural category: Modeling

Maturity: Research

SCA relationship: Compatible

What is described?

A3S proposes using UML for modeling and non-functional verification of radio architectures and application requirements, specifically surrounding the SPS, devices, and interfaces to higher-level processing. The A3S modeling tool aims at generating a mapping of the SW operations on the HW devices with some pre-verification of coherence and performance, which would then be followed by autogeneration using some other tool.

How would it improve code portability if adopted widely?

The A3S approach will reduce the cost of bringing up an implementation on a new platform. It will enable HW providers to check platform performance and so to provide platforms well-adapted to wider class of waveforms. It will also enable exchange of models between industrial partners to check portability in between various contexts.

How does it compare to widely used alternative approaches?

The UML approach of A3S will be based on OMG SWradio DSIG profile and is aiming at being fully compatible with the SCA.

What is the level of technology maturity, standardization, commercial acceptance?

Although what A3S describes is the result of laboratory experimentation, they are based on commercially available tools. Therefore, the overall technology is quite mature and accepted, if not in the precise context of the HAL RFI.

Other issues to consider:

Although the A3S proposal does not directly address the question posed by the RFI, it does suggest a design and verification approach that could provide significant value. Work on a standard for a HAL solution should not preclude design and verification processes as described in this response.

Waveform Description Language (1)

Authors: Mark Freeman

Organization: QinetiQ

Architectural category: Modeling

Maturity: Research

SCA relationship: Conflict

What is described?

QinetiQ describes the Waveform Description Language project which they managed on behalf of the UK MOD. This was a two phase project, phase 1 a specification language and phase 2 an implementation environment. Phase 2 was implemented by BAE and is described from their perspective in the autogeneration section.

This page describes phase 1. Phase 1 was based on the Berkeley Ptolemy II modeling environment. Its goals were:

- Mitigate errors arising when textual waveform standards are translated to implementation artifacts.
- Enable unambiguous yet unconstrained capture of waveform requirements.
- Provide a clear mapping to a variety of design flows to enable the (autonomous) generation of hardware and software code.
- Employ a standard graphical syntax to facilitate the visual interpretation of waveforms.
- Enable simulation of the model to test and verify functionality.

How would it improve code portability if adopted widely?

WDL will compress waveform specification/design/implementation cycles. It provides an executable specification/reference model which forms the skeleton of the implementation design. This provides maximal levels of consistency between the specification and implementation phases, and thereby reduces cost.

How does it compare to widely used alternative approaches?

At present waveform specifications are expressed in ambiguous natural language form. Compared to this, a rigorous machine-readable representation offers obvious advantages.

What is the level of technology maturity, standardization, commercial acceptance?

Early research. The Ptolemy II environment requires substantial expertise to exploit effectively. Therefore it may not be an ideal platform for a widely used design tool.

Other issues to consider: Proprietary content, etc.

UK MOD decided to terminate the specification language effort at the end of phase 1 and switch to an implementation effort in phase 2. There is no ongoing work on the specfication language described here at this time.

Signal Processing System Description

Authors: Dr. Cyprian Grassmann

Organization: Infineon Technologies AG Corporate Research Systems Technology

Architectural category: Modeling

Maturity: Research

SCA relationship: Conflict

What is described?

This response explores general concepts for system description of signal processing sub-systems.

Signal processing sub-system description needs to be sufficiently abstract but at the same time contains enough information which can guide mapping decisions and therefore substantially ease the process of compilation, in particular to the next generation of parallel architectures.

In this context, the following system design rules are recommended:

- Signal processing data flow and control flow should be separated.
- A library of functional primitives for signal processing subsystems should be defined.
- The interface to functional primitives should be open and well documented so that the library of functional primitives should be easily extensible.
- The description of the primitives itself should be hardware independent and should express data and task level parallelism explicitly.
- Latency and throughput requirements should be part of the system description.
- A representative set of functional signal processing blocks should be specified.
- The system description can be simulated to allow for easy test and verification of algorithms.

How would it improve code portability if adopted widely?

These recommendations should provide benefits in:

- simplifying different mappings.
- simplifying the analysis by compilers or mapping tools.

How does it compare to widely used alternative approaches?

Current design approaches are lacking a hardware independent system description, including timing, throughput and silicon area requirements.

What is the level of technology maturity, standardization, commercial acceptance?

The proposal is a concept and parts of the concept are already in-house implemented.

Other issues to consider: Proprietary content, etc.

As these general concepts do not exhibit any run time interface or component model, they conflict with the object-oriented SCA approach.

Simulink framework for sharing specifications and requirements

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Modeling

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Use of Simulink in large complex projects as a common language for teams to use to capture and communicate their designs. Whether it is the communication of the specification from the customer to the prime contractor, or collaboration among the prime contractor's interdisciplinary design teams, or the specification of subsystems and components to suppliers, accuracy and completeness are critical. A common framework such as Simulink provides the necessary structure while allowing the flexibility and extensibility to include as much design detail as necessary. Reference waveforms and test environments such as channel models can be packaged with the specification as part of a Simulink model. Custom libraries can easily be created to support specific projects.

How would it improve code portability if adopted widely?

Simulink is a hierarchical block diagram environment suited to capturing requirements starting at the system level and moving down into subsystems and individual components. Encapsulating the design into blocks and subsystems is a natural way to define APIs to enhance portability. Having an interactive model serve as the specification for a system greatly reduces the risk of misinterpretation of implementation details, and thereby reduces expected porting cost.

How does it compare to widely used alternative approaches?

Paper based specifications typically contain errors and omissions, but a far greater limitation is that they are static. It is impossible to probe the specification for answers to "what if" questions. Using a Simulink model as a specification allows the recipient to probe the specification at various levels to gain clearer understanding of the requirements.

What is the level of technology maturity, standardization, commercial acceptance?

Simulink was released around 1993. Since then, the breadth, scale and efficiency of the development environment, simulation capabilities and code generation have been continuously improved.

Simulink and related products are in widespread use in the automotive, aerospace, electronics, semiconductor industries and other embedded electronics applications. In the automotive industry, leading manufacturers are using Simulink to specify requirements to their OEM suppliers.

Other issues to consider: Proprietary content, etc.

Simulink and related products are proprietary technology of The MathWorks. These products and features cannot be implemented or utilized without a license from The MathWorks. However, APIs for third parties or end-user customization are well documented, stable and publicly available.

Simulink for hardware-independent end-to-end system models

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Modeling

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Using Simulink to create a hardware-independent end-to-end system model. This enables engineers to simulate and test the complete system behavior, and refine their designs before implementing them. The Simulink family of products can model systems that contain analog and digital signal flow and control flow behavior. For complex, multi-rate signal processing systems, the DSP Blockset provides libraries of basic signal processing components such as filters, transforms, and buffers along with powerful math operations such as statistics, parametric estimation, and matrix operations. For waveform design, the Communications Blockset provides libraries of common modem functions such as modulation, coding, and interleaving.

How would it improve code portability if adopted widely?

A single model that is hardware-independent creates the framework for design portability. Once the model is validated, partitioning and implementation of the design on a particular hardware platform can proceed using automatic code generation, customized code generation, or manually. In each case, the design team can verify the implementation against the model using it as a test harness. All these benefits reduce cost and hence improve portability.

How does it compare to widely used alternative approaches?

Other approaches to creating hardware-independent end-to-end system models include computer programs in various languages and mathematical models that are typically summarized and communicated in paper reports. Such reports at best provide snapshots of the design and are not the most effective way to enable design portability. By creating and validating a model in Simulink, designs can be captured and communicated in a way that allows the implementers to interact with the design and ask important "what if" questions. Simulink provides a natural way to express the concurrency critical to many signal processing applications.

What is the level of technology maturity, standardization, commercial acceptance?

Simulink was released around 1993. Since then, the breadth, scale and efficiency of the development environment, simulation capabilities and code generation have been continuously improved. Simulink and related products are in widespread use in the automotive, aerospace, electronics, semiconductor industries and other embedded electronics applications.

Other issues to consider:

Simulink and related products are proprietary technology of The MathWorks. These products and features cannot be implemented or utilized without a license from The MathWorks.

Data types and APIs for fixed-point numerics, simulation, and code generation

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Modeling

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Since 1998, the Fixed Point Blockset has equipped Simulink with fixed-point data types and numeric processing. This supports modeling of fixed-point algorithms or for use during the conversion of floating-point algorithms into fixed-point.

Related products, such as the DSP Blockset and the new Fixed Point Toolbox for MATLAB, leverage this technology by offering fixed point processing in many core algorithms including filters, transforms, and linear algebra.

How would it improve code portability if adopted widely?

The fixed point representation embodied in MathWorks Fixed Point Toolbox and Simulink Fixed Point products provides a common implementation-neutral framework for expressing fixed point data type characteristics for signal processing applications. Included are word size, radix position, signed/unsigned, rounding modes, saturation/wrap effects, real/complex, slope/bias, etc. Also, a foundation of analysis capabilities are provided, such as characterization of when dynamic range limits have been hit or exceeded, investigating tradeoffs in precision, etc. These tools assist design engineers and reduce the cost of the fixed-point conversion during a porting process.

How does it compare to widely used alternative approaches?

Other ways to approach fixed-point numerics is to use an extension to C or C++, say SystemC or DSP-C or another programming language extension. The execution performance achieved by these solutions during simulation is faster than the baseline execution of M-code in MATLAB. However, third parties such as Catalytic offer direct compilation of M-code into C, for the express purpose of simulation acceleration.

What is the level of technology maturity, standardization, commercial acceptance?

The Fixed Point Blockset for Simulink was introduced in 1998. In 2004, a new release of the Filter Design Toolbox will build upon the Fixed Point Toolbox to offer a much broader range of fixed-point design and analysis capabilities for digital filters. The pre-release software has gone to hundreds of active sites worldwide. Several Third-Party vendors are incorporating or building upon this framework for existing and new product offerings, both in MATLAB and in Simulink.

Other issues to consider: Proprietary content, etc.

Simulink, MATLAB and related products are proprietary technology of The MathWorks. These products and features cannot be implemented or utilized without a license from The MathWorks.

Open platform for verification, validation and test.

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Modeling

Maturity: Widely used product

SCA relationship: Compatible

What is described?

An open platform for hardware and software IP verification and validation and system testing. Specific products that exploit the open platform include:

- *Link for Code Composer Studio* – provides programmatic control and co-simulation with Texas Instruments' Code Composer Studio from MATLAB and Simulink as well as real-time data exchange and debugging of software running on Texas Instruments DSPs.
- *Link for ModelSim* – provides co-simulation and testbench automation from MATLAB and Simulink to verify HDL designs in Mentor Graphics' ModelSim software.
- *Instrument Control Toolbox* – enables live data acquisition and control of analytical instruments from MATLAB

Many 3rd parties and customers have used the open APIs available in MATLAB and Simulink to automate testing and verification using other commercially available and in-house tools.

How would it improve code portability if adopted widely?

- Provides testbench portability – maintain bit- and cycle-accurate models that conform to waveform standard, but are hardware independent.
- Permits early design and validation of test cases and maintains a "golden reference" design with testbenches that can be re-used as implementation platforms or system partitioning changes.
- Validated waveform models can be used with PC-based instrumentation to perform sophisticated testing of components and final product validation.

How does it compare to widely used alternative approaches?

Traditionally, test benches need to be re-implemented at each stage of development to be compatible with the tools used at that stage. Maintaining testbenches with the validated system model can reduce time and the risk of errors inherent in the current process.

Integration of waveform models with PC-based analytical instruments can significantly reduce the cost of equipment required for testing software-defined radios.

What is the level of technology maturity, standardization, commercial acceptance?

The Link for Code Composer Studio has been commercially available for three years. The Link for ModelSim has been available for three months. The Instrument Control Toolbox has been available for two years.

Other issues to consider:

MATLAB, Simulink, blocksets and related products are proprietary technology of The MathWorks. These products and features cannot be implemented or utilized without a license from The MathWorks.

Environment

These responses propose standard features for the operating environment of software components inside the SPS. They reduce porting cost by reducing the changes required when porting a software component across different platforms.

Portable Components using Containers for Heterogeneous Platforms

Authors: Murat Bicer

Organization: Mercury Computer Systems, Inc.

Architectural category: Environment

Maturity: Research

SCA relationship: Compatible

What is described?

Recommended standardized interfaces on non-GPP processing elements between algorithmic cores and an infrastructure that provides isolation from the platform, board and chip specifics. An algorithmic implementation needs to be independent of the platform in which it was initially implemented (the algorithmic implementation could be a Signal Processing Subsystem (SPS) as addressed by this RFI or any other intellectual property).

How would it improve code portability if adopted widely?

By using this approach, code written for non-GPP (General Purpose Processor) hardware can directly be ported to a new platform, provided the new platform has the implementation of the "container" (i.e. the specification that is proposed by this response). By defining and standardizing the component-to-container interactions appropriate to DSP and FPGA containers, the performance "tax" of a full CORBA implementation is eliminated while still retaining the portability advantages.

How does it compare to widely used alternative approaches?

Using standardized APIs, especially for inter-device communication, is relatively new to engineers developing FPGA and DSP solutions. Any such standards will likely be seen as constraining – much like initial uses of the C language found it constraining when compared to assemblers. Standard interfaces in these domains tend to be defined in a technology-specific way. Driving such definitions based on coexistence and compatibility with software-centric approaches enables greater portability of the complete component-based application and more flexible technologies choices over different deployment constraints and over time as technologies change.

What is the level of technology maturity, standardization, commercial acceptance?

The basic technique of standardizing FPGA interfaces to interact well with software components has been the subject of IR&D at Mercury for some time. They believe enough experience has been gained to propose appropriate interfaces that map from an IDL subset to local container interfaces for DSPs and FPGAs.

Other issues to consider:

The proposed technology is proprietary since no standards have been defined, but Mercury is willing to contribute to a standardization process.

Digital IF Specification

Authors: Tansu Demirbilek

Organization: Mercury Computer Systems, Inc.

Architectural category: Environment

Maturity: Research

SCA relationship: Compatible

What is described?

The Digital IF standard specifies hardware interfaces for digital receivers and exciters, protocol specification for data transfer, control, management and synchronization, as well as software interfaces for communication between tuners and digital processors.

This specification is an initiative to address the I/O standardization problem in the SDR. The intent is for the standard to be able to abstract and control any I/O interaction through the standardized interfaces. The SCSI standard for disk drives/subsystems constitutes a nice analogy for the Digital IF specification, since its focus also broadened in time to allow usage in any system regardless of bandwidth requirements. What SCSI is to disks, Digital IF is to tuners/exciters digital input/output.

How would it improve code portability if adopted widely?

By using this approach, data communication between digital processing components and data transfer/receive components can be standardized. This means that digital processing components can be designed and implemented without considering the semantics of data movement between processing components and digital converters.

How does it compare to widely used alternative approaches?

Existing solutions on the market that address the digital IF interface problem use proprietary technology. This specification proposes an open standard aimed to generate a common understanding between tuner vendors and digital processor vendors.

What is the level of technology maturity, standardization, commercial acceptance?

The Digital IF specification is being initiated by Mercury with participation from other digital tuner vendors. Their plan is to submit this work to a standardization body.

Other issues to consider:

None

OMG SWRadio specification

(formally known as “Specification for PIM and PSM for Software Radio Components”)

Authors: Tansu Demirbilek

Organization: Object Management Group (OMG)

Architectural category: Environment

Maturity: Standard in development

SCA relationship: Close

What is described?

The OMG SBC DTF has defined and voted on (as of April) a Software Radio specification which defines radio infrastructure facilities that can be utilized in developing waveforms. The response goes on to describe the three major chapters of the OMG spec – a UML profile for software radio, a processor independent model (PIM), and a processor specific model (PSM) for CORBA IDL. The UML profile defines a language for modeling a software defined radio. The PIM provides a model of software system behaviour and API’s along with example component definitions. The PIM is based in part on APIs developed by the System Interface Working Group of the SDR Forum. The PSM defines a mapping of the PIM onto CORBA IDL.

How would it improve code portability if adopted widely?

The OMG response indicates that the SWRadio specification provides:

- a component based software architecture that allows easy technology insertion
- standard APIs for interacting with waveform components
- through the PIM/PSM paradigm, a model that allows porting to any specific platform.

How does it compare to widely used alternative approaches?

The OMG standard is based on and compatible with the SCA.

What is the level of technology maturity, standardization, commercial acceptance?

The OMG SWRadio specification is in the final submission stage and was voted on for acceptance at the April OMG meeting. During the finalization task force phase, all OMG Architecture Board issues will be resolved and a proof-of-concept implementation will be developed.

Other issues to consider: Proprietary content, etc.

The OMG is a widely regarded standards body in the software community. Any standard issued by them would have been vetted by a large number of OMG member companies. Proprietary content (i.e. content available only to OMG member companies) may be a concern until the final specification is approved and released.

Hardware Abstraction Layer Definition Within SCA

Authors: Bill Lawrence / Patrick Jeffers

Organization: General Dynamics (http://www.gdds.com)

Architectural category: Environment

Maturity: Research

SCA relationship: Close

What is described?

This response explores the waveform portability gains associated with the application of Quality of Service (QoS) guarantees across all waveform interfaces to all devices of the SDR.

The QoS definition encompasses not only processing capabilities, but process threading, process boundary, memory partitioning, process/thread relative priority & scheduling, and data transport QoS definitions between the processing entities as well.

As modeled in UML, the QoS takes the form of additional classes (TimeValue, ClockInterrupt, ActionExecution, QoSvalue, Instance, etc.) and inherits modifications to existing classes (e.g. Resource, Device, etc.) which provide a richer set of interfaces and associated semantics across the entire SCA model enabling communication of QoS requirements for waveforms.

In order to facilitate a more homogeneous environment for waveform execution, this response endorses the application of an ORB into the domain of the SPS (e.g. on DSPs).

How would it improve code portability if adopted widely?

A consistent set of QoS related interfaces incorporated into both the processor RTOS process/threading/memory management set and the transport layer should allow waveforms to constraint the associated devices with their needed throughputs and deadlines.

How does it compare to widely used alternative approaches?

Current SDRs manage the real-time constraints of their supported waveforms by “tweaking” either their waveforms, or their platform software or both.

Waveforms designed using the QoS constructs should explicitly declare their real-time constraints in terms of the real-time constructs provided. This should allow a priori determination of potential real-time matching between the candidate waveform and target SDR platform.

What is the level of technology maturity, standardization, commercial acceptance?

This response leverages two existing OMG specifications:

- the OMG UML Profile for Schedulability, Performance and Time [1],
- the OMG Real-Time CORBA (Dynamic Scheduling) specification [2].

Additional work will need to be performed for interfaces which do not leverage CORBA (e.g. FPGAs & Security boundaries) as QoS constraints will propagate into the implementations.

A Rose model which leverages these specifications is currently under development by GDDS.

Other issues to consider:

Application of the QoS technique will require adaptation across the entire SCA, through the addition of application deployment constraints to the SCA, in particular the Device type.

Platform Abstraction Layer

Authors: Graham McKenzie

Organization: Celoxica Ltd.

Architectural category: Environment

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Two interface layers: the Platform Abstraction Layer (PAL) which provides a standard interface between an FPGA-based application and physical resources such as I/O or memory, and the Data Stream Manager (DSM) which provides an abstraction layer between processing elements.

How would it improve code portability if adopted widely?

The basic concept of the two abstraction layers will undoubtedly lead to improved code portability. Although the response lacked enough detail to say for certain, it seems that in both cases transactions are abstracted to common, natural methods of interaction. For instance:

- PAL would provide a standard means to access memories attached to an FPGA.
- PAL would provide a standard means to move data in and out of an I/O device.
- DSM would provice a standard means to move data between processing resources utilizing various mechanisms (i.e. DMA).

How does it compare to widely used alternative approaches?

The proposal closely matches the approach taken by many different organizations.

What is the level of technology maturity, standardization, commercial acceptance?

PAL support for Celoxica's own hardware products has been available for 2 years and is now in its third major release. Over this time the API has been refined and the mechanisms for supporting different classes of I/O has been improved to provide a stable platform for application development.

DSM is now in its second major release and provides support for Virtex II Pro (Power PC) and Microblaze as well as MS Windows hosts to the Celoxica RC2000 Virtex II based boards.

Other issues to consider: Celoxia has suggested that the solution be standardized and have offered to participate in any standardization activity. This should remove any serious proprietary content issues so long as they do not assert licensing rights.

In their response Celoxia mentioned a number of times their access to the PAL and DSM via Handle-C. Without more details it is not possible to tell how easily this can be migrated to other FPGA application tools/languages. This is a possible issue as Handle-C is not a universally recognized FPGA development approach.

As is typical of such an approach, there are shortcomings. The most fundamental shortcoming is that the general patterns or behavours of data movements are often difficult to represent abstractly. For instance, one FPGA application may want to access locally attached memory as a single, large logical memory block, while another may want to independently access individual memory components. In the case of an I/O device there may be features that are unique to the particular device that may be difficult to represent in a general, abstract way.

Hardware Abstraction Layer Software Development Library

Authors: Lee Pucker, Kevin Maier, Chris Brand

Organization: Spectrum Signal Processing.

Architectural category: Environment

Maturity: Widely used product

SCA relationship: Compatible

What is described?

Spectrum Signal Processing's *quic*Comm™ software development library acts as a hardware abstraction layer in that it provides a common API for communicating between disparate processing elements in a heterogeneous processing environment. *quic*Comm, fundamentally, supports all of the elements outlined in RfiQ9.

How would it improve code portability if adopted widely?

*quic*Comm supplements the POSIX standard by providing a simple API focused on the movement of data streams between disparate hardware components. This significantly decreases the effort required to move functionality between the different processing elements, allowing the application to be more easily optimized for the deployment platform. It also supports technology insertion in future evolutions of similar architectures.

*quic*Comm is implemented as a thin layer, maximizing performance while minimizing overhead associated with a structured protocol. In addition, *quic*Comm can support different levels of optimization allowing trade-offs to be made between performance and features (such as error checking).

How does it compare to widely used alternative approaches?

Conceptually, *quic*Comm is similar to the Berkeley sockets model of IP addresses and ports, but with additional control over the routing of data. In the Berkeley sockets model, the address of a specific interface is identified as Network.Sub-Network.Node:Port, whereas in the *quic*Comm API a model of System.Board. Node:Channel is utilized. Note that the *quic*Comm interface does not constrain the implementation.

What is the level of technology maturity, standardization, commercial acceptance?

*quic*Comm has been available from Spectrum Signal Processing since July 2000, and evolved from Spectrum Signal Processing's Alib API, which had been in use since 1997. It has been ported to support TI C62x and C64x DSPs, Motorola MPC74xx and 82xx PowerPCs, IBM 405 PowerPCs, VirtexII and VirtexII Pro FPGAs as well as several different single-board computers and development hosts. In addition, *quic*Comm supports a number of different transport mechanisms including PCI, RapidIO, VME, Solano (a proprietary transport mechanism similar to RapidIO), Race/Race++, as well as direct links between processing devices using interfaces such as the EMIF on a TI DSP or direct connection to an FPGA. *quic*Comm supports the ACE/TAO CORBA ORB and fully supports the Software Communications Architecture (SCA) version 2.2.

A version of Spectrum's SDR-3000 platform incorporating our *quic*Comm API and the SCA has been selected by JTeL as a representative hardware set for the JTRS.

Other issues to consider:

While the *quic*Comm API is published, the implementation is currently proprietary.

API for component library supporting both simulation and run-time execution

Authors: Alex Rodriguez

Organization: The MathWorks, Inc.

Architectural category: Environment

Maturity: Widely used product

SCA relationship: Compatible

What is described?

This product consists of a set of API definitions for signal processing components and a compliant C implementation of each one. This supports the design, simulation, and deployment of high-speed signal processing subsystems. The target library is generally not used directly by a Simulink user. It is automatically utilized during simulation and code generation by the tool chain.

The target library broadly spans control systems and digital signal processing, including basic math, filters, transforms, matrix/linear algebra, and many other algorithm areas. It contains over a thousand APIs defined in ANSI C headers, each category of functionality containing functions for multiple data types, real/complex data types, frame-based (vector) and scalar implementations, single- vs. multi-channel implementations, etc.

How would it improve code portability if adopted widely?

The component libraries are target-neutral API definitions that lead to the creation of larger sets of portable intellectual property (IP). This IP is still specializable to a particular target via underlying C-callable APIs that are vendor-optimizable.

How does it compare to widely used alternative approaches?

This offering may compete with other predefined libraries, such as the VSIPL library, the Intel IPL, or other C-programming APIs focusing on signal processing applications. Similarities to VSIPL include C-callable APIs, a methodology surrounding each to make the libraries easier to integrate and use, and an open, vendor-optimizable API for gaining target-specific optimizations.

The strengths of the Simulink solution include:

- Simulink graphical models automatically generate code that calls to these library functions, and Simulink applies global optimizations for determining which functions to call.
- There is no "minimum" set of APIs that must be implemented on the platform. Simulink will automatically provide generic ANSI-C implementations for all functions not otherwise supplied/optimized by a vendor.
- High degree of (automatic) usage by existing customers implies significant testing of interfaces and their utility in practical applications.

What is the level of technology maturity, standardization, commercial acceptance?

The run-time library solution has been widely in use in commercial products for more than 5 years, implying a broad level of testing and refinement.

Other issues to consider:

MATLAB, Simulink, blocksets and related products are proprietary technology of The MathWorks. These products and features cannot be implemented or utilized without a license from The MathWorks.

Definition of Signal Processing Subsystem (SPS) Hardware Abstraction Layer

Authors: Christian Serra, Eric Nicollet

Organization: Thales Communications S.A. – 160 Bd de Valmy – 92704 - Colombes - France.

Architectural category: Environment & Control

Maturity: Early Product

SCA relationship: Close

What is described?

This response describes a set of interfaces to provide SPS waveform portability and interactions with SCA compliant parts running on HLPS (GPP implemented). SPS software runs on DSP & FPGA. On the SPS side, the following interfaces (APIs) are identified:

- real-time HLPS Communication API, from SCA *pushPacket()*
- Management Device APIs, from SCA *Resource* and *SignalError*
- real-time Radio Devices APIs (Transceiver, Audio ACR (Acquisition Restitution),…)
- Processing Services APIs (SP Libraries, SP Intellectual Property (IP) Blocks,…)
- Operating System (OS) APIs, a POSIX subset

On the HLPS side, the following APIs are identified which comply with the SCA "Logical Devices" concepts : (GPP implemented)

- real-time SPS Communication, from SCA *pushPacket()*
- SPS DSP and FPGA Logical Device, from SCA *Executable / Loadable Device*
- SPS Resource, from SCA *Resource*
- Radio Logical Devices (Transceiver, Audio ACR,…), from SCA *Aggregate Device*

How would it improve code portability if adopted widely?

The proposed approach clarifies the architectural relationships inside the SPS architecture, clarifies the relationships between the HLPS architecture and the SPS architecture, and extends Object Oriented (OO) concepts inside the SPS. This proposed approach is also compliant with state-of-the-art SP Waveform Development Environment (WDE).

How does it compare to widely used alternative approaches?

This approach extends the SCA concepts to the SPS functions, without applying the cumbersome SCA Operating Environment architecture (ORB based) inside a real-time domain. It also extends the current APIs definition for both modular FPGA implementation and multiprocessing SPS architecture.

What is the level of technology maturity, standardization, commercial acceptance?

The interfaces are currently defined and implementations under development.

Other issues to consider:

Proprietary in-house implementations

Control/Other

These responses describe methods for configuring and controlling the SPS in an abstract fashion. They reduce porting cost by reducing the changes required in the rest of the system when the SPS changes.

High level abstraction of signal processing elements

Authors: Craig Dolwin (Toshiba Research Europe Ltd)
Siegfried Walter (Alcatel)
Jörg Brakensiek, Bernd Steinke (Nokia Research Centre)
Klaus Moessner (The University of Surrey)

Organization: E2R (An EU sponsored research consortium "End-to-end reconfigurability")

Architectural category: control

Maturity: Research

SCA relationship: Conflict

What is described?

Work package 4 of the E2R project is proposing a configuration interface to the signal processing system in a wireless terminal (UE) or basestation(RAN)/access point. The signal processing system encompasses functions such as RF components, Analog Front End, Digital Front End, Baseband Processing and Source Coding.

The configuration interface has two parts, runtime and description. At the runtime level, the authors propose mechanisms to support the exchange of reconfiguration parameters and software to UE and basestation/access point and network entities. At the description level, E2R proposes to exploit XML to describe the desired configuration. This Configuration Description will define the data flow between sets of processing functions and the associated constraints such as timing deadlines and performance criteria.

How would it improve code portability if adopted widely?

A common public configuration interface will hide the underlying proprietary hardware from the rest of the system.

How does it compare to widely used alternative approaches?

Other approaches require all possible configurations to be pre-defined when the hardware is designed. This severely limits the number of configurations supported and creates substantial difficulties when the network attempts to reconfigure both ends of a communication link. Existing proprietary reconfiguration approaches limit interoperability and restrict the use of reconfiguration in heterogenous networks.

What is the level of technology maturity, standardization, commercial acceptance?

The E2R project began work in January 2004.

Other issues to consider:

As a European Union sponsored collaborative research project, E2R is likely to have significant influence on future European standards.

A method of specifying FPGA control interfaces using XML

Authors: Vince Kovarik

Organization: Harris Corporation

Architectural category: control

Maturity: Research/Early product

SCA relationship: Close

What is described?

The Harris response describes an approach to specifying the control interfaces for FPGA-based waveform implementations using an interface mapping XML.

How would it improve code portability if adopted widely?

The approach assists code portability by allowing the interface to be specified rapidly for low-level VHDL waveform implementations. When the VHDL changes, the HLPS code that controls it need not change because all specific register addresses are loaded automatically from the XML file.

How does it compare to widely used alternative approaches?

Other approaches for supporting flexible interfaces for VHDL implementations in FPGAs have often been focused around the development of specific library interfaces. This can present problems in the long term because they can require extensions and modifications as new waveforms are delivered. The approach described does not require modification to a common library or linking into the software radio code base. New waveforms can be rapidly hosted by developing the XML file that describes the mappings.

What is the level of technology maturity, standardization, commercial acceptance?

The approach has been tested in the laboratory and demonstrated SCA compliant waveform loading, configuration, and control. It is being refined and evolving into an internal standard for achieving SCA compliance for FPGA-based waveforms and is still undergoing internal development and refinement.

Other issues to consider:

This will be the basis of a Harris SCA compliant radio and appears to be proprietary.

Device-centric SDR solutions and the Software Communications Architecture

Authors: John D. Bard

Organization: Space Coast Communication Systems, Inc.

Architectural category: Other

Maturity: Research

SCA relationship: Close

What is described?

A proposed hierarchy of abstraction layers, inspired by the abstraction layers in a PC which have allowed decades of technology insertion and high application portability. These are as follows.

1. Microelectronics data sheet.
2. Memory-mapped abstraction
3. Target specific low level driver
4. OS specific high level driver
5. Language bindings for high level driver
6. Radio service abstraction
7. Low level application software

Each layer hides differences in the lower level components. For example, the memory-mapped abstraction will make chips from different manufacturers look similar. However, the layers should not be fully abstract. A higher layer may need to specify precise behavior by lower layers to meet its performance or functionality requirements.

One of the principal goals of this hierarchy of abstractions is relative independence of the application software from critical timing paths within the radio devices. Additional layers of abstraction can continue all the way up to the CORBA/XML level.

How would it improve code portability if adopted widely?

An effective hierarchy of standard abstractions would improve portability at all levels of the system.

How does it compare to widely used alternative approaches?

In the more commonly proposed abstraction approaches, the layers strive for full abstraction. For example, radio services software might bundle several lower level functions into a well-defined higher level function like "ACQUIRE_PSK". In practical experience generic functionality like this does not offer the high level of reuse one would expect. There are two reasons for this: 1) Generic algorithms might not offer the performance required for a particular modulation scheme and 2) fielded legacy applications rarely stick to the "text book" and often perform some kind of customization either for performance or intellectual property reasons.

What is the level of technology maturity, standardization, commercial acceptance?

This is an observation and a suggestion only.

Other issues to consider:

None.