Rapid Prototyping of Application Specific Signal Processors

RASSP

Process - Architecture - Framework

Rapid Prototyping of Application–Specific Signal Processors (RASSP)

RASSP Model Year Architecture Specification Volume I

Introduction to Model Year Architecture

Version 1.0

September 5, 1996

ADVANCED TECHNOLOGY LABORATORIES

LOCKHEED MARTIN

Table of Contents

1. INTRODUCTION 1

1.1. Background 1
1.2. Purpose 1
1.3. Scope 2
1.4. Document Outline 2

2. REQUIREMENTS 3

3. MODEL YEAR ARCHITECTURE OVERVIEW 6

4. MYA SPECIFICATIONS 11

4.1. Vol. I – Introduction to Model Year Architecture 11
4.2. Vol. II – Hardware Architecture Element Specification 11
4.3. Vol. III – Standard Virtual Interface Specification 11
4.4. Vol. IV – Standard Virtual Interface Specification Appendices 11
4.5. Vol. V – Software Architecture Element Specification 12
4.6. Vol, VI – Model Year Architecture Re–Use Library Element Specification 12

5. SUMMARY 12

Appendix I. Acronyms used in RASSP Model Year Architecture Specification 15

1. OVERVIEW

1.1. Background

Traditional design of embedded military signal processors has hinged on providing the best possible performance for a particular application or problem domain. The results were highly customized solutions that frequently used proprietary or non–standard interfaces and required long development cycles, which incurred high non–recurring engineering (NRE) costs. Upgradability, while sometimes addressed in the initial design, still required significant NRE costs because of the design's custom / proprietary nature.

The RASSP program goal is to develop an approach that will drastically improve the process by which embedded signal processors are designed. Such improvement will result in a fourfold reduction in both design cycle for initial design and fielding as well as for upgrading existing designs, and life–cycle costs (LCC). This improvement will be achieved through a methodology that stresses reusing hardware and software elements in conjunction with an architectural approach that encourages and facilitates reusability and upgradability through the adoption of open interface standards. This architectural approach is termed the Model Year Architecture (MYA). The term "Model Year Architecture" implies a design approach in which architectural elements (hardware/software) are upgraded on an evolutionary basis, and are made available as reuse elements to be employed to design new, or upgrade existing, RASSP systems. Based on this approach, a system designed with the architectural elements of a given model year is typically an evolutionary improvement over a system from the previous model year, not a completely redesigned system.

1.2. Purpose

The purpose of the RASSP Model Year Architecture Specification is to consolidate and communicate the latest developments in the Lockheed Martin RASSP Model Year Architecture definition. This document, *RASSP Model Year Architecture Specification – Version 1.0*, represents the first formal release of the MYA specification and coincides with the completion of the primary phase of the RASSP program. This specification supercedes information contained in preceding versions of the *RASSP Model Year Architecture Working Document*, and has been refined through three previous iterations during the course of the RASSP program. This version of the MYA Specification differs from previous versions of the MYA Working Document in that the specifications for the four key components of the Model Year Architecture — hardware architecture elements, the Standard Virtual Interface, software architecture elements, and reuse library elements — have been broken out as separate volumes of the MYA Specification. Except for the Standard Virtual Interface specification, the information covered in these volumes was previously included in the MYA Working Document. This volume of the MYA Specification presents an overview of the Model Year Architecture approach and introduces its constituent components.

The refinement process through which the MYA Specification has and will continue to occur can be described by The Risk Driven Expanding Information Model, or RDEIM, which is being used

to describe the RASSP Design Methodology. The RDEIM concept is a useful and appropriate model for the development of the Model Year Architecture definition since the refinement process is based on identifying risks and issues with the baseline definition, addressing those risks and issues having the highest priority and in so doing, reducing those risks while increasing the level of information available to refine and update the definition. The progression through several cycles or iterations has gradually expanded the information content and refined the MYA framework definition, and simultaneously reduce the level of risk associated with it

Risk abatement and information expansion have occurred through investigative activities as part of the Architecture Task. Issues have been identified in previous drafts of this document and at subsequent working group meetings and have been resolved through further refinement of the architecture definition as well as efforts on the Model Year Architecture Verification subtask. This subtask was a major investigative activity which has been formally defined as part of the Architecture Task to address and reduce risks associated with definition of the architectural constructs of the MYA framework through modeling and simulation exercises.

1.3. Scope of the Model Year Architecture

The Model Year Architecture scope includes the following elements:

1) An approach for scalable interconnection of modules or elements that perform signal processing functions. Such elements include standard programmable processors (general purpose or DSP), custom programmable processors or accelerator hardware, special–purpose processors (e.g. SIMD processors), sensors, and memory modules.

2) An approach to interface a signal processor subsystem to other platform subsystems.

3) An approach to architectural–level library component definition that facilitates rapid insertion of upgraded architectural components without requiring redesign of other system components.

4) A modular software architecture incorporating standard application programming interfaces that facilitates rapid insertion of upgraded software elements (operating systems, drivers, application libraries) with minimal module interaction and interdependencies. The modularity of the software architecture will also accommodate hardware upgrades by confining changes to hardware specific software modules.

5) The use of open interface standards wherever possible; that is, interface standards that are both commercially accepted and not proprietary to ensure interoperability between components.

1.4. MYA Specification Outline

The Model Year Architecture Specification consolidates the four specifications of the primary components of the Model Year Architecture, along with an overview of the Model Year approach

and rational. This document, Volume I, serves as an introduction to the Model Year Architecture and to the constituent components of the overall specification. Section 2 summarizes the rationale and driving requirements for the RASSP Model Year Architecture. Section 3 presents an overview of the RASSP Model Year Architecture approach. Section 4 provides an introduction to each of the remaining volumes of the MYA Specification. Section 5 presents a summary of the Model Year Architecture concept.

The remaining volumes of the MYA Specification are organized as follows: Volume II – RASSP Hardware Architecture Element Specification; Volume III – RASSP Standard Virtual Interface Specification; Volume IV – RASSP Standard Virtual Interface Specification Appendices; Volume V – RASSP Software Architecture Element Specification; and Volume VI – RASSP MYA Re-Use Library Element Specification.

2. REQUIREMENTS

The drivers for RASSP signal processor architecture definition result from the requirements imposed on signal processors to meet mission-critical processing needs as well as a 4X reduction in development cycle and life-cycle costs (LCC). Development cycle and LCC requirements have fueled the need to address low-cost technology insertion, upgradability, and extensibility. This section discusses the requirements imposed on RASSP signal processors and their impact on signal processor architecture and design. These requirements form the basis for the RASSP Model Year Architecture development and are encompassed in the categories of *Development Cycle Reduction, Life Cycle Support, Scalability, Heterogeneity, Flexible Interfaces, Modular Software, Testability,* and *System Retrofits.*

Development Cycle Reduction

The concept of the Model Year Architecture revolves around a new paradigm for signal processor design that is based on the concept of successive refinement through a series of short design cycles. This paradigm will replace the traditional one of developing optimized point designs that result in long development cycles and difficult and/or costly upgrades. The successive refinement approach supports rapid delivery of initial prototype systems based on existing technology, possibly at the expense of some performance, to allow end users to evaluate and test the system. The trend of rapid performance improvement in commercial technology enables the system to be successively refined, upgraded with new technology, and debugged through several design cycles on a Model Year basis. This results in the delivery of a system (in the same amount of time) with more functionality, fewer flaws, and less cost. The desire to incorporate multiple design cycles into a development program is the basis for requiring a 4X reduction in current design cycle times.

Life Cycle Support

Life cycle support is an essential requirement for RASSP signal processors and is one of the prime motivators of the RASSP philosophy. The cost and logistics associated with the life cycle of a fielded signal processor are at least as important as the initial savings in development cost and

schedule. Life cycle support must be designed into the signal processor from the start. This includes aspects of both Design For Test (DFT) to support expedient low-cost field maintenance and diagnostics, and designing for model year upgradability that will extend the useful life cycle of the signal processor. The following requirements follow directly from the needs of life cycle support and development cycle reduction.

Scalability

Experience has shown scalability — along with size, weight, and power — to be an extremely important requirement for military signal processor systems. Changing scenarios and the need for new counter-measures often require increased functionality over the system's life cycle. Furthermore, the same basic system may be deployed with different amounts of functionality depending on the intended mission environment. All these factors require a system that is highly scalable to minimize NRE, recurring, and life cycle costs.

The system-level scalability requirement in turn imposes a requirement on the interconnection of system elements. To meet a high degree of scalability, system elements must be interconnected to impose few, if any, restrictions on the number of elements to be interconnected (from a very small to very large number), with no adverse affects on performance. The interconnection scheme must also support low-latency realtime communication.

Heterogeneity

The diverse domains of signal processing and the wide range of performance requirements for different systems impose an implicit requirement to support heterogeneous designs. This implies the use of several different processing element types in a single system to implement functions they are most suited to. For example, application of adaptive filter weights to a high-speed data stream may be done most cost effectively by a specialized vector processor; computing the weights to apply may incorporate some heuristic aspects and be updated at a fairly modest rate, and may be done most cost effectively on a general-purpose processor. Heterogeneous designs must support the incorporation of the following elements:

1) *General-Purpose Processors*

2) *Digital Signal Processors (DSP)*

3) *Special-Purpose Processors (e.g. SIMD processors)*

4) *Custom Hardware Accelerators*

5) *Shared Memory Nodes*

Flexible Interfaces

A signal processor is a component of a larger system, and as such it must interface to other components or subsystems. RASSP signal processors must provide an approach to interface to the following types of subsystems:

1) *Loosely-coupled processor subsystems (data processors, consoles / workstations)*

2) *Mass storage systems (disk, tape)*

3) *Ancillary equipment (receiver / transmitter control, navigation, etc.)*

In addition to subsystem interfaces, RASSP processors must provide a standard approach for sensor interfacing that can accommodate the typically custom nature of sensors without placing too many restrictions on them. Such sensors include receivers, video sources, audio sources, etc., that provide digital outputs within the defined scope of RASSP. Analog signal processing, conditioning, and digitizing (A/D) are considered outside the scope of RASSP, although interfacing to the digital side of A/D converters is considered within the scope.

Modular Software

Experience has shown software development and maintenance to be a very costly part of signal processor design, primarily as a result of the lack of both code reuse and standard programming languages. The fact that signal processors have historically been custom designs with custom assembly or microcode, which were hand-coded at that level to achieve maximum performance, meant that software developed for one design usually could not be reused on another. If a system needed to be upgraded with a faster processor, the assembly or microcode would not necessarily be compatible with the original, which resulted in a large recoding effort.

RASSP signal processors must be supported by a software approach that is modular, readily reusable, and upgradable with minimal module interdependencies and interactions. Software developed for a particular design must be reusable for subsequent designs, particularly common signal processing functions that share widespread applicability to many domains. To support hardware upgrades, software must be easily portable, which dictates using standard programming languages and standard application programming interfaces. Modularity must be implemented to minimize interdependencies and interactions, which allows underlying operating systems, services, and library implementations to be upgraded with little or no impact on application software.

Testability

RASSP signal processors must incorporate a test strategy as an integral part of the design to ensure a high-quality product while maintaining the aggressive goals of design cycle, non-recurring, and life cycle cost reduction. An ad hoc or inadequate treatment of testability will manifest itself by severely impacting component testing, system integration, and test aspects of the design cycle, as well as field maintenance and diagnostics during its life cycle.

RASSP processors must be able to detect and isolate faults with high probability; for high-reliability systems they must also reconfigure in realtime. To achieve these goals, Built-In Test (BIT) must be included at the processor system, chassis, board, and multichip assembly levels. Hierarchical test and maintenance busses must be included to control test assets at lower levels and to report test results up to the system level. To meet the rapid prototype goals, COTS components will likely be used, but chips and components that support boundary scan and Built-In–Self–Test (BIST) are preferred over ones that do not. When new components are designed, DFT features must be included.

System Retrofits

In addition to new systems, RASSP signal processors may be retrofitted into previously designed non–RASSP (legacy) systems, which means providing support to incorporate RASSP signal processors into non–RASSP systems.

In summary, the RASSP Model Year Architecture specification provides an approach to designing signal processors which fully addresses these stated requirements. This specification will be integrated into the RASSP design methodology to provide the framework for implementing signal processor designs that meet these requirements.

3. MODEL YEAR ARCHITECTURE OVERVIEW

This section provides an overview of the Model Year Architecture framework which has been defined to address the requirements presented in Section 2. The architectural constructs to be described rely heavily on the mutual support of hardware and software elements.

Model Year Architecture characteristics needed to support the stated requirements are:

– *Modular and scalable open architecture*

– *Communications via non–proprietary, standardized interfaces*

– *Leverage of commercial technology (components, boards, etc.) whenever possible*

– *Support for heterogeneity*

– *Incorporation of custom components*

– *Modular software architecture*

– *Reusability of components*

– *Support for low–cost hardware/software upgrades for continuous product improvement*

In defining a Model Year Architecture framework, one must consider how it will be used to benefit a signal processor design. What does this framework provide with respect to the RASSP Methodology? Figure 1 depicts a User's View of the Model Year Architecture. The user or system designer has the task of designing a system to solve a particular application problem. What is available to the user is the problem definition and the RASSP Methodology. These two items are insufficient to design a signal processor that meets all RASSP requirements. The user could design a signal processor, but it would most likely be a custom solution to optimally fit the specific problem at hand, which would lengthen the initial design cycle (Note: RASSP does not preclude customization, but allows its use within constraints). Also, there are no guarantees regarding the degree of upgradability and facility for technology insertion that the user may consider and the overall result may tend toward the old point design paradigm. The Model Year Architecture framework is integrated into

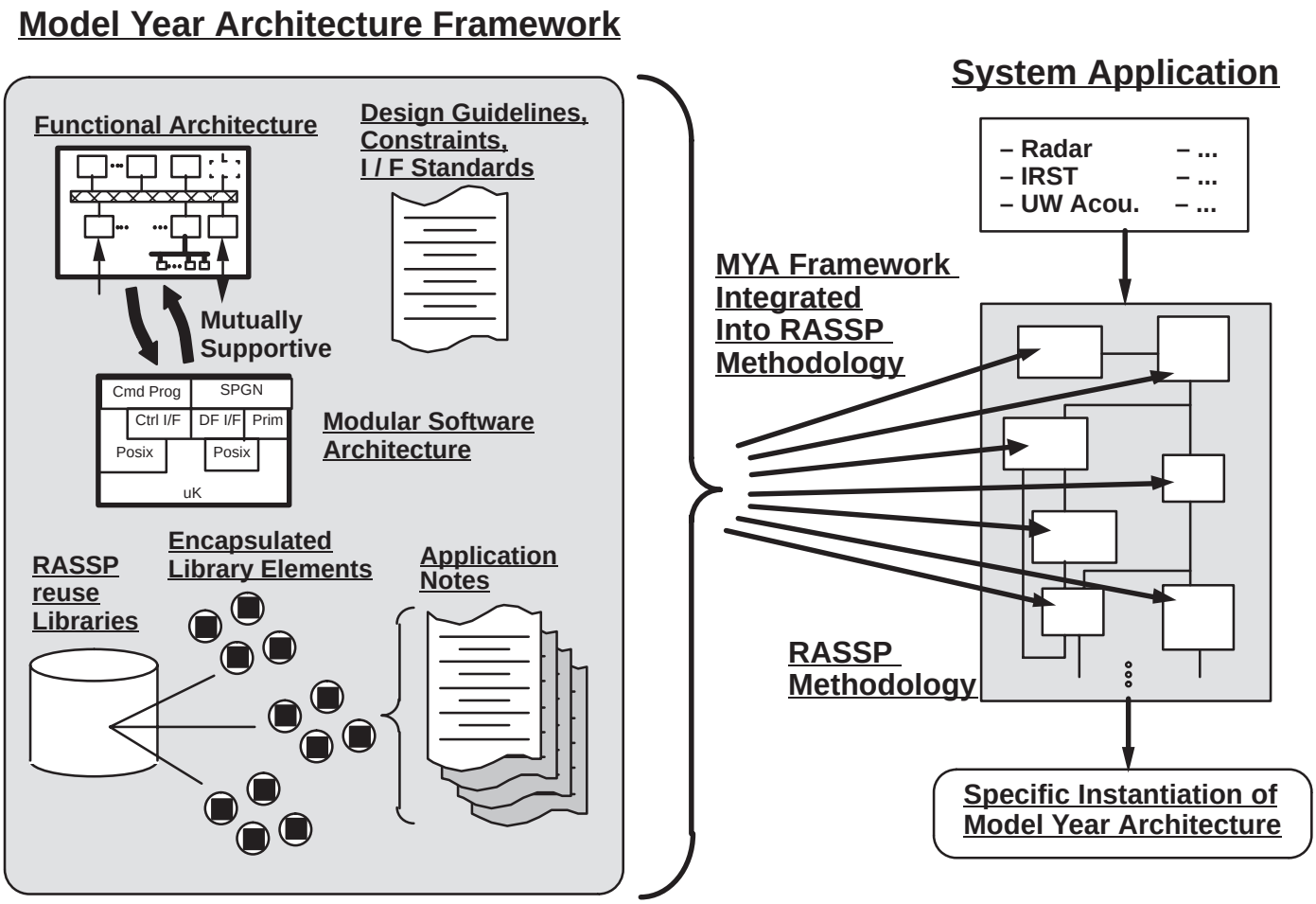

Figure 1 User's View of Model Year Architecture

the RASSP Methodology to provide additional resources and constraints to the design process. This will enforce a structured approach to ensure that the design incorporates all the required features of a Model Year signal processor.

The Model Year Architecture framework is comprised of several components that affect how a user designs a signal processor. These components, which impact the RASSP Methodology in architecture development, hardware/software reuse library management, hardware synthesis, and target software generation, are formally specified in the remainder of the Model Year Architecture Specification. The following descriptions are an overview of the five major components that comprise the Model Year Architecture Framework.

Functional Architecture

The Functional Architecture defines the necessary components at the architectural level and the manner in which their interfaces must be defined to ensure that the resulting architecture design

is upgradable and facilitates technology insertion. As such, the Functional Architecture is a starting point for developing an architecture for an application–specific problem, *not* a detailed instantiation of an architecture. The diversity of application domain requirements precludes specifying instantiation details at this level. Experience has shown the need to significantly tailor architectures for different application domains. While every design should start with the Functional Architecture, specific architectural details needed to serve the application problem at hand, such as what type(s) of and how many processing elements (PEs) to use, the best interconnect topology for the problem domain, the need for shared memories, etc., will be developed as part of the Architectural Selection portion of the RASSP Methodology.

Encapsulated Library Components

Although the RASSP Functional Architecture defines architectural level components and their interface definitions, the embodiment of a Model Year Architecture lies in the reuse libraries which support it. The RASSP reuse library element structure is defined with a VHDL modeling hierarchy with three levels of representational abstraction:

Level 1: Token–Based Performance Models – Timing–only behavior to support high level architectural tradeoffs.

Level 2: Abstract Behavioral Models – Full timing and behavior of multi–component electronic modules to support module–level trade–off analysis.

Level 3: Detailed Behavioral Model – Timing and behavior of individual integrated circuits for low–level design and verification.

Above these three abstraction levels is assumed to be an executable specification, also in VHDL, which describes the combined hardware and software performance and functionality of the system. In addition to the three levels of representational abstraction, the reuse library also contains a structure relating to the physical hierarchy of the reuse elements. Together, the physical hierarchy and representational abstraction provide a hierarchy of *views* of a RASSP system.

To support the Functional Architecture, reuse library elements must be encapsulated at the architectural level. Encapsulation refers to additional structure or *wrappers* added to otherwise "raw" library elements to support the Functional Architecture framework and ensure library element interoperability and upgradability. The concept of reuse requires encapsulated library components at the architectural level (PEs, internal interconnect interface elements, shared memories, subsystem and sensor interface elements) to ensure that the developed signal processor incorporates the required model year characteristics. The Model Year Architecture library encapsulation requirement is one of the key drivers in developing and maintaining reuse libraries. Incorporated within the reuse libraries are application notes that the designer can use to properly apply each component. Section 4 describes how encapsulation supports the Functional Architecture, and Section 5 specifies how encapsulations are created.

The concepts of encapsulation, the use of standard interfaces, and hiding of implementation details are used by Object Oriented Design (OOD), which has been successfully applied to software development. RASSP is extending the concepts of OOD to signal processor architecture design, en-

compassing both the hardware and software aspects to support the Model Year concept. In essence, an architectural library element such as a processing element can be thought of as an object (or more precisely, an object class which becomes an object upon instantiation) as depicted in Figure 2 which possesses:

1) *A restricted but standard interface (the functional interface)*

2) *A defined functional capability or set of **methods** implemented via a combination of hardware (represented by VHDL models) and software accessible through the object's member functions (standard software library functions).*

3) *Encapsulation to hide implementation details from the user and to limit the propagation of changes resulting from upgrades.*

The implementation of the object's functionality requires a co–dependency of its hardware and software aspects, necessitating hardware–software codesign of such object–oriented architectural library elements. Operating systems require configuration to take advantage of whatever hardware resources are available to support its functionality. Likewise, application libraries may need configuration or optimization dependent on the hardware implementation.

A concise definition of the various object classes and their attributes for reuse is contained in MYA Spec. Vol. VI – MYA Re-Use Library Element Specification.

Modular Software Architecture

As stated earlier, hardware and software elements must be mutually supportive to successfully implement the framework needed to design Model Year Architectures. As such, a modular software architecture extends the Model Year Functional Architecture into the software domain. The definition of standard requirements for operating system functionality, services, and interfaces, the use of standard application programming interfaces, and separable hardware–dependent software modules (e.g. I/O drivers) with standard interfaces, will encapsulate the functionality of software modules, separating and hiding implementation details from the designer. Such software modules will be defined to support the Functional Architecture elements. This form of encapsulation will support plug–and–play reuse and model year upgradability of operating system kernels, external services, application libraries, as well as support hardware upgrades. This renders the application software virtually portable to the underlying hardware and software platforms and is a primary goal of the Model Year Architecture.

Open Interface Standards

Open interface standards must be used to ensure interoperability between components. This means hardware and software interface standards that are both commercially accepted and not proprietary. Open standards for RASSP interfaces will be incorporated into hardware and software library components. As new / improved standards evolve, these can be incorporated as new encapsulated library components to facilitate their interoperability with previously defined components, offering Model Year upgrade capability.

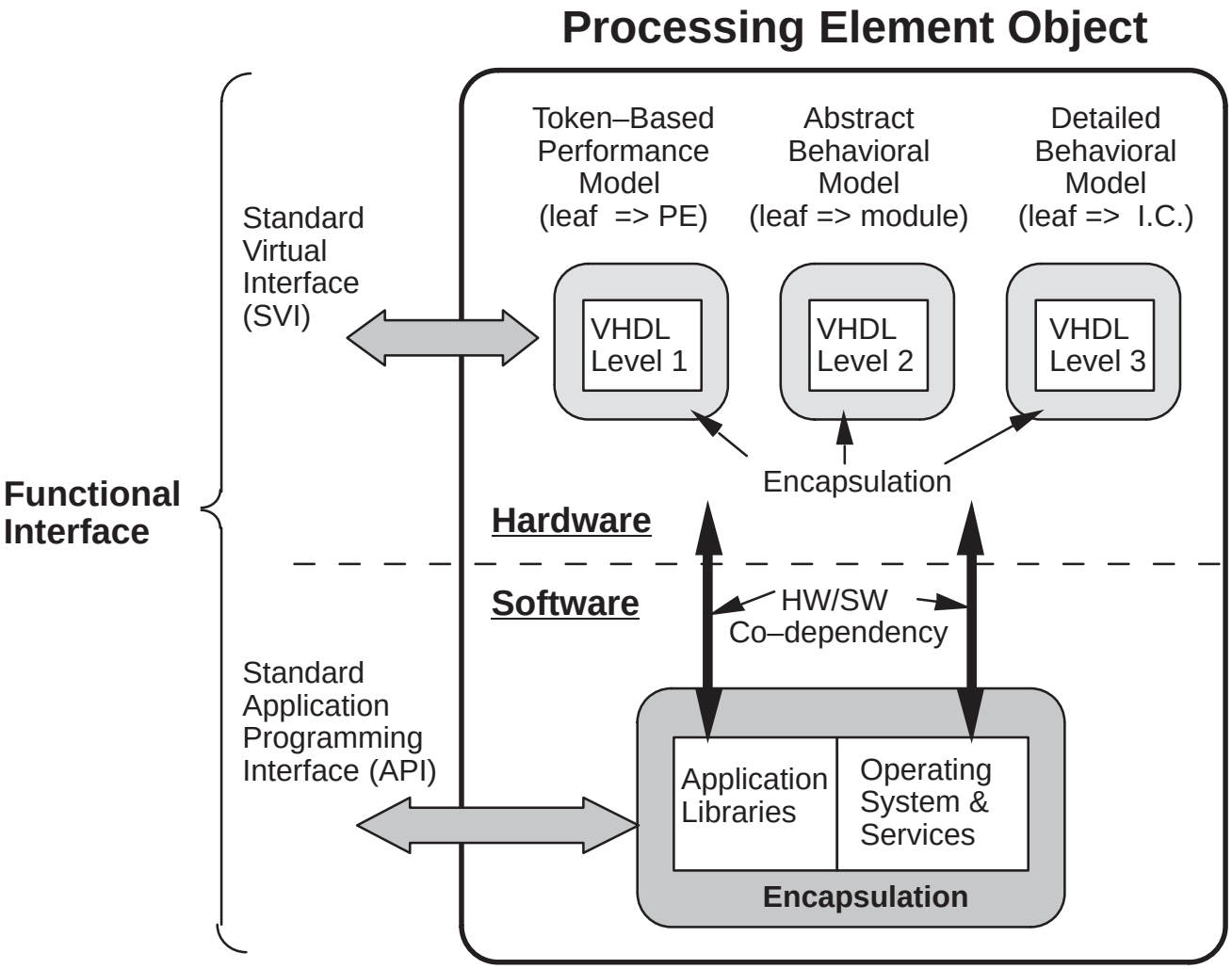

Figure 2 Processing Element Object Example. Object Oriented Design of Architectural library elements facilitates rapid design and Model Year Upgrades.

Design Guidelines and Constraints

The Model Year Architecture Framework also provides a set of design guidelines and constraints for general architectural development, such as how to properly use the functional architecture, general use of encapsulated libraries, and most importantly, procedures to encapsulate new library components. Such design guidelines and constraints will be incorporated into the RASSP design methodology.

4. MYA SPECIFICATIONS

The following six volumes of specifications comprise the RASSP MYA Specification:

4.1. Vol. I – Introduction to Model Year Architecture

This document, Volume I, serves as an introduction to the Model Year Architecture and to the constituent components of the overall specification.

4.2. Vol. II – RASSP Hardware Architecture Element Specification

The purpose of the RASSP Hardware Architecture Element Specification document is to convey a formal specification for hardware architecture elements which comprise the RASSP Model Year Architecture (MYA) Functional Architecture. The Functional Architecture defines the necessary components at the architectural level and the manner in which their interfaces must be defined to ensure that the resulting architecture design is upgradable and facilitates technology insertion. Adherence to this specification will ensure the creation of architectural elements, and the design of specific architectures which include them, which will provide the RASSP MYA features of architectural element reuse, interoperability, and facilitated upgradability.

In particular the specification: describes the MYA Functional Architecture and offers design guidelines; describes the unique interface approach devised for assembling architectural level components in the functional architecture to create RASSP processor systems, and provides interface guidelines; and describes the requirements to fully implement architectural level component VHDL models as RASSP reuse library elements.

4.3. Vol. III – RASSP Standard Virtual Interface Specification

The purpose of the RASSP Standard Virtual Interface Specification is to convey a formal specification for the Standard Virtual Interface (SVI), a technology–independent functional interface which facilitates hardware upgrades and technology insertion at the architectural level. This specification will be used when designing and modeling architectural reuse library elements to ensure interoperability among those various library elements. In particular, the SVI specification contains: a concise definition of the signals, functional timing, and protocol of the SVI.

4.4. Vol. IV – RASSP Standard Virtual Interface Specification Appendices

The purpose of the RASSP Standard Virtual Interface Specification Appendices are to serve as concrete aids to encapsulation designers. In particular, the appendices contain templates for SVI encapsulations, specific VHDL encapsulation examples; and guidelines for library element designers to ensure that resulting encapsulations are interoperable with one another and that they are synthesizable.

4.5. Vol. V – RASSP Software Architecture Element Specification

The purpose of the RASSP Software Architecture Element Specification is to support a methodology that simplifies developing high–performance, realtime DSP applications. This methodology allows the application developer to easily describe and implement signal processing algorithms and control the execution of these algorithms. It supports the Model Year concept by allowing the application to be developed in a platform–independent fashion. And it provides predictable deterministic response to all services provided. The architecture is defined to support the programming of signal processing applications using a data flow graph approach, to provide high–level command program and data flow graph development, and POSIX to support a generic operating system service interface. These layers are built on top of a realtime microkernel supplied with the model year hardware platform. The RASSP Application Programmer Interface (API) implemented using the data flow methodology and POSIX will remain fixed throughout model years, ensuring continued and easy porting of existing applications.

4.6. Vol. VI – RASSP Model Year Architecture Reuse Library Element Specification

The purpose of the RASSP Model Year Architecture Reuse Library Element Specification is to provide all information required to completely define MYA reuse library elements, and will be used to create templates for the architectural reuse library database. A concise definition of the MYA reuse element composition hierarchy will be defined that will specify all required and optional components of each architectural reuse element object class and their dependencies.

5. SUMMARY

This document has presented an introduction and overview of the Lockheed Martin RASSP Model Year Architecture specification. The definition of the Model Year Architecture is based on requirements for signal processor design that have evolved from experience in signal processor system design, as well as the additional requirements imposed by the RASSP philosophy to address a 4X reduction in development cycle and life–cycle costs, technology insertion, upgradability, and extensibility. The characteristics of the Model Year Architecture are that it: 1) must be modular, scalable, and open (non–proprietary), 2) should incorporate non–proprietary standard interfaces wherever possible, 3) should leverage commercial technology whenever possible, 4) must provide a way to incorporate custom components when necessary, 5) must facilitate reusability of components, and 6) must provide support for low–cost hardware and software upgrades for continuous product improvement.

The Model Year Architecture definition is a framework of five major components that will be integrated into the design methodology:

1) **A Functional Architecture** that provides a starting–point for developing an application–specific architecture and the architecture selection process.

2) **Encapsulated Library Components** which implement the Functional Architecture by adding encapsulation structures to "raw" architectural library elements to implement standard Functional Architecture constructs. Such constructs ensure component interoperability and upgradability. Encapsulated library components are supported at multiple levels of VHDL hierarchy based, in part, on the VHDL DID. Application notes provide an aid to the designer in proper application of each component.

3) **Modular Software Architecture** to provide an extension of the Functional Architecture into the software domain to support upgrades of operating system kernels, external services, and application libraries. The modular software architecture will also provide support for hardware upgrades by confining associated software changes to hardware–specific software modules.

4) **Open Interface Standards** that are commercially accepted to further ensure interoperability between components and to ensure wide availability of commercial components and support.

5) **Design Guidelines and Constraints** for general architectural development, such as how to properly use the Functional Architecture, general use of encapsulated libraries, and most importantly, procedures for generating the encapsulation for new library components.

The purpose of the Model Year Architecture framework is not to specify particular topologies or PE types to use because these are highly dependent on the specific application at hand. This framework is viewed as an enabler for the design methodology to produce specific instantiations of signal processors that incorporate Model Year characteristics.

The structure of the reuse library components at the architectural level contains the essence of what makes a resulting architecture a Model Year architecture. Using Object Oriented Design techniques to define the various Functional Architecture constructs provides a convenient and natural approach for developing upgradable architectures as well as designing and maintaining the reuse library elements themselves. Library element object classes include both hardware and software portions which are co–dependent, each of which contributes to the object's total behavior. The hardware and software portions are encapsulated to hide implementation details from the user, limit propagation of changes resulting from upgrades, and to ensure interoperability. The encapsulation implements a standard functional interface that is accessible to the user: a hardware portion called a Standard Virtual Interface, and a software portion that is a standard Application Programming Interface (API). By limiting access to this interface, upgradability and technology insertion is greatly enhanced. In addition, the formal but judicious use of layering within both the hardware and software architecture plays an additional role in defining library element object classes and where various interfaces are most appropriate.

The use of open interface standards will further ensure interoperability between and reuse of various signal processor elements, and will help ensure multiple sources and reasonable costs for compatible interface components, eliminating the dependence on sole–source proprietary components.

The Model Year concepts of reusability, upgradability, and technology insertion extend to the Model Year Software Architecture which must support a methodology to simplify the development of high performance real time digital signal processing applications. A standardized Application Programming Interface (API) based on Real–Time POSIX to support application control and a data flow algorithm representation based on NRL's PGM specification will be provided. This API will be supported by a set of primitive libraries as well as a specialized run–time system to control and execute multiple data flow graphs on a multi–processor system. The foundation of the software architecture leverages commercial real–time microkernel technology to provide the basic services needed to support the higher level services provided to the application programmer.

Appendix II. Acronyms used in RASSP Model Year Architecture Specification

A/D	Analog–to–Digital
AEP	Application Environment Profiles
ANSI	American National Standards Institute
AP	Arithmetic Processor
API	Application Programming Interface
ARPA	Advanced Research Projects Agency
ASIC	Application–Specific Integrated Circuit
ASW	Anti–Submarine Warfare
ATM	Asynchronous Transfer Mode
BIT	Built–in Test
BIST	Built–in Self–Test
CCITT	International Telegraph and Telephone Consultative Committee
CPU	Central Processing Unit
DFG	Data Flow Graph
DFT	Design For Test
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
EDAC	Error Detection and Correction
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EMP	Electromagnetic Pulse
EMSP	Enhanced Modular Signal Processor
EW	Electronic Warfare
FDDI	Fiber Distributed Data Interface
FFT	Fast Fourier Transform
FPGA	Field–Programmable Gate Array
HIPPI	High Performance Parallel Interface
HLL	High Level Language
HW	Hardware
I/F	Interface
I/O	Input/Output
ID	Identifier
IEEE	Institute of Electrical and Electronic Engineers
IRST	Infrared Search and Track
ISO	International Standards Organization
JIAWG	Joint Integrated Avionics Working Group
JTAG	Joint Test Action Group

LRM	Line Replaceable Module
MIPS	Millions Of Instructions Per Second
NGCR	Next Generation Computer Resources
NRE	Non–Recurring Engineering
NRL	Naval Research Laboratory
OOD	Object Oriented Design
OSI	Open System Interconnect
OSM	Off–Board Service Manager
PCB	Printed Circuit Board
PE	Processing Element
PGM	Processing Graph Method
PLD	Programmable Logic Device
POSIX	Portable Operating System Interface for UNIX
RNI	Reconfigurable Network Interface
RTL	Register Transfer Level
R/T	Real Time
RASSP	Rapid Prototyping of Application–Specific Signal Processors
RRTS	RASSP Run–Time System
SCI	Scalable Coherent Interface
SCSI	Small Computer System Interface
SONET	Synchrounous Optical Network
SPGN	Signal Processing Graph Notation
SRAM	Static Random Access Memory
SVI	Standard Virtual Interface
SW	Software
TBD	To Be Determined
TI	Texas Instruments
TM	Test and Maintenance
TMC	Test and Maintenance Controller
TPM	Target Processor Map
VITA	VME International Trade Association
VME	Versa Module Eurocard
VXIbus	VME Extensions for Instrumentation Bus
XDR	External Data Representation