A Novel Specification and Design Methodology Of Embedded Multiprocessor Signal Processing Systems Using High-Performance Middleware

Randall S. Janka[1] and Linda M. Wills[2]

[1] Georgia Institute of Technology, Georgia Tech Research Institute,
Atlanta, GA 30332-0856 USA
randall.janka@gtri.gatech.edu
[2] Georgia Institute of Technology, School of Electrical and Computer Engineering,
Atlanta, GA 30332-0250 USA
linda.wills@ee.gatech.edu

Abstract. Embedded signal processing system designers need to be able to prototype their designs quickly and validate them early. This must be done in a manner that avoids premature commitment to the implementation target, especially when that target includes costly COTS parallel multiprocessing hardware. A new specification and design methodology known as MAGIC enables the designer to move from an executable specification through design exploration and on to implementation with minimal loss of specification and design information by leveraging compuation middleware (VSIPL) and communication middleware (MPI). Maintaining such information is a quality known as "model continuity," which is established using the MAGIC specification and design methodology.

1 Introduction

Embedded signal processing system designers need to be able to prototype their designs quickly and validate them early. This results in quicker time to market as well as early detection of errors, which is less costly. There is tremendous complexity in the specification and design of these systems even when we restrict the technology space to commercial-off-the-shelf (COTS) multiprocessing (MP) hardware and software. We need a way to manage this complexity and accomplish the following goals:

- Enable the designer to quickly evaluate and validate design prototypes.
- Reduce and manage the level of detail that needs to be specified about the system in order to make sound decisions at each stage of the design process.
- Allow the design space to be explored *without* committing too early to a particular technology (hardware platform).
- Enable constraints identified and derived in one stage to be applied consistently in other stages of the design process.

In other words, we need to be able to benchmark and validate in early stages (at the appropriate level of detail and without premature commitment) – a process we call "virtual benchmarking" [1]. We also need to carry information gained (constraints and design rationale) through to later stages, a quality known as "model continuity." We have developed a new methodology to do this by exploiting computation and communication middleware that are emerging as standards in the embedded real-time COTS multiprocessing domain.

2 The Need for Model Continuity in Specification & Design Methodologies

The process of designing embedded real-time embedded multiprocessor signal processing systems is plagued by a lack of coherent specification and design methodology. A canonical waterfall design process is commonly used to specify, design, and implement these systems with COTS MP hardware and software. Powerful frameworks exist for each individual phase of this canonical design process, but no single methodology exists which enables these frameworks to work together coherently, i.e., allowing the output of a framework used in one phase to be consumed by a different framework used in the next phase.

This lack of coherence usually leads to design errors that are not caught until well into the implementation phase. Since the cost of redesign increases as the design moves through these three stages, redesign is the most expensive if not performed until the implementation phase. We have developed design rules and integrated commercial tools in such a way that designs targeting COTS MP technologies can be improved by providing a coherent coupling between these frameworks, a quality known as model continuity.

The basic information flow of a COTS MP specification and design (SDM) methodology is shown in **Fig. 1**. To appreciate how our SDM establishes model continuity, we first illustrate how model continuity is missing in today's COTS MP methodologies, as shown in **Fig. 2**. Currently, constants such as filter coefficients can be passed from MATLAB .m files into a CASE SDM or a simpler vendor software development environment, but that is the only link from the requirements specification and design specification to the implementation phase in the whole design process. Not having an executable requirements model and a channel for passing it to the design analysis phase leads to *model discontinuity*, which is the total absence or minimal presence of model continuity.

3 The MAGIC Specification and Design Methodology

We have developed and prototyped a new SDM which we call the MAGIC[1] SDM [2]. The means of accomplishing model continuity using the frameworks we chose for the MAGIC SDM is illustrated in **Error! Reference source not found.**. Solid boxes are

[1] MAGIC–Methodology Applying Generation, Integration, and Continuity.

documents or frameworks. Dashed boxes are aggregates of frameworks that contain executable specifications or the design analysis environment. Solid lines are automated channels, where system model information can be passed between frameworks without manual intervention. Dashed lines are semi-automated channels where some human intervention is required to move system model information between frameworks.

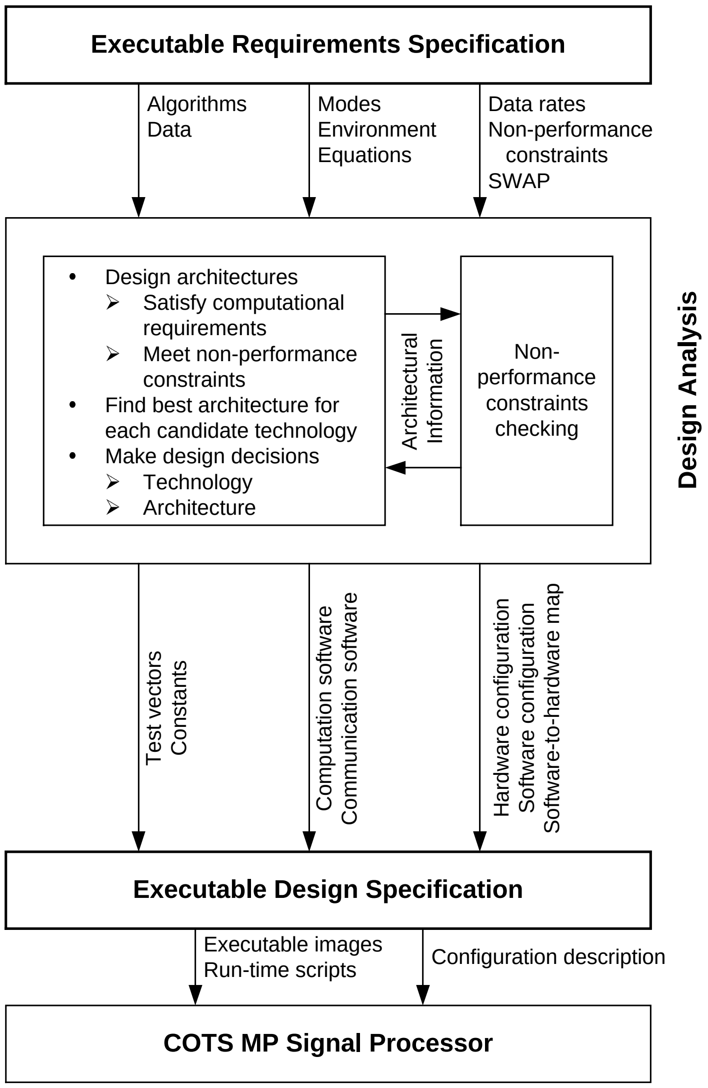

Fig. 1. Basic flow of information needed to support model continuity.

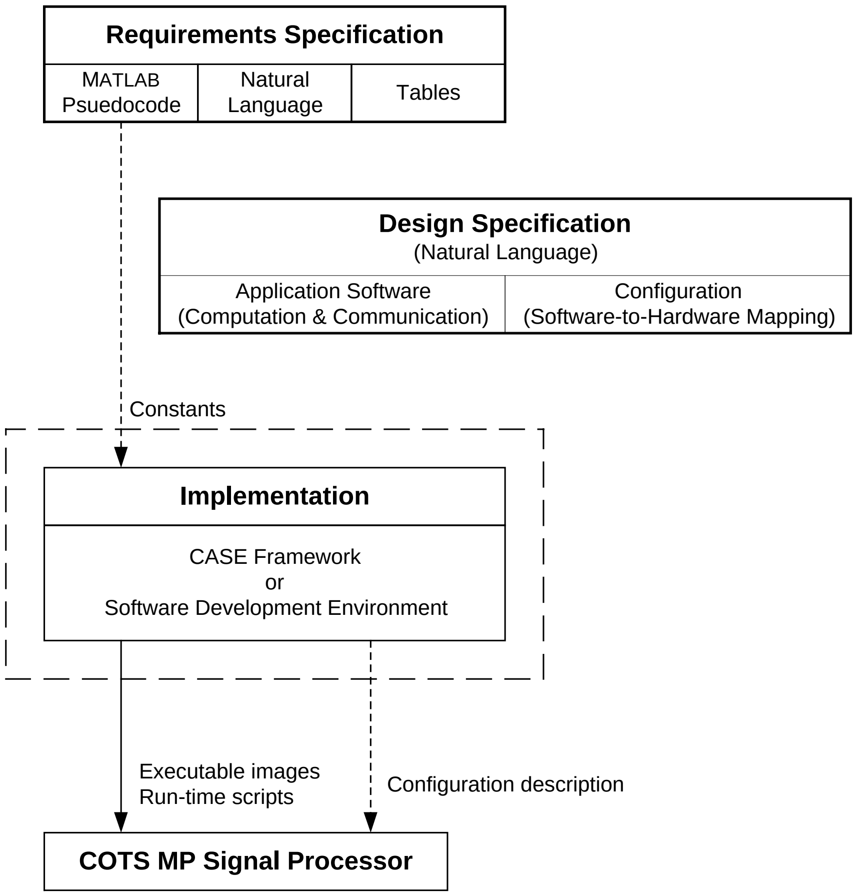
Requirements Specification
MATLAB Psuedocode
Natural Language
Tables
Design Specification
(Natural Language)
Application Software (Computation & Communication)
Configuration (Software-to-Hardware Mapping)
Constants
Implementation
CASE Framework or Software Development Environment
Executable images
Run-time scripts
Configuration description
COTS MP Signal Processor

Fig. 2. How model continuity is currently lacking in current COTS MP SDM.

The executable workbook was fundamental in providing model continuity between specification and design. It was created using Excel with links created between worksheets that contained data (benchmarks, reliability statistics, form factor constraints, etc.) and models (benchmark conversions, process estimates, latency estimates, etc.). The data link to Simulink[2] was manual; architectural parameters were computed in Excel and then implemented in Simulink by hand since Simulink does not support scaling for parallelization. VSIPL[3] (computation middleware) and MPI[4] (communication middleware) functions were "generated" using our code generation rules and entered into our executable workbook. Once in our workbook, we could compute

[2] Simulation and rapid prototyping framework from The MathWorks.
[3] Vector Scalar Image Processing Library–an open-standards API for computation.
[4] Message-Passing Interface–an open-standard API for multiprocessing and parallel processing communication. Its real-time cousin is "MPI/RT."

token delays to be used in eArchitect[5] for performance modeling. We would iterate this process for other candidate architectures.

We created channels of model continuity between specification and design with the implementation specification. When we decided upon an architecture, we could run Simulink and tap process outputs, dumping them into the MATLAB workspace where we could save them for testing the implementation. VSIPL and MPI code that we generated is available for use in the form of the inner-loop functions and parameter arguments. When design analysis is complete and we have made design decisions, our performance model provides the hardware configuration, software process definition, and software-to-hardware mapping.

4 Model Continuity via Middleware

Model continuity is achieved in large part through the use of middleware for computation and communication. Open standards-based middleware supports computation and communication software portability, which means that middleware written for one vendor's hardware should run on another vendor's platform. Consequently, middleware code that constitutes the inner-loop software implementation can be used for different vendors' platforms for design analysis using performance modeling. Critical to making the use of middleware a strong thread of model continuity is the autogeneration of middleware code, since automating the generation of software by a framework that is correct in specification reduces the chance of error in the design and implementation.

A code generator such as Simulink's Real-Time Workshop that could generate middleware for computation using VSIPL, MPI for communication, and/or MPI/RT for communication and control will produce code for both design and implementation. The generated middleware can be used to quantify process delays in the performance model framework and as the core for signal processing implementation application software.

Our reasons for choosing VSIPL and MPI are very similar to our reasons for choosing the frameworks discussed above. They are stated here in order of importance with the most important reason stated first:

- Acceptable performance–These middlewares deliver high-performance because they are tightly integrated with the vendors' computation and communication libraries.
- Standards-based–Since all the COTS MP vendors in our domain space support these middleware and actively participate in their standardization processes, frameworks that generate VSIPL and MPI code will be consumable by all of the hardware vendors' SDEs considered in the design phase.
- COTS–They are now becoming commercially available and therefore stable and supported.

[5] Performance modeling framework from Viewlogic that supports multiprocessing and high-speed interconnections such as RACEway and Myrinet.

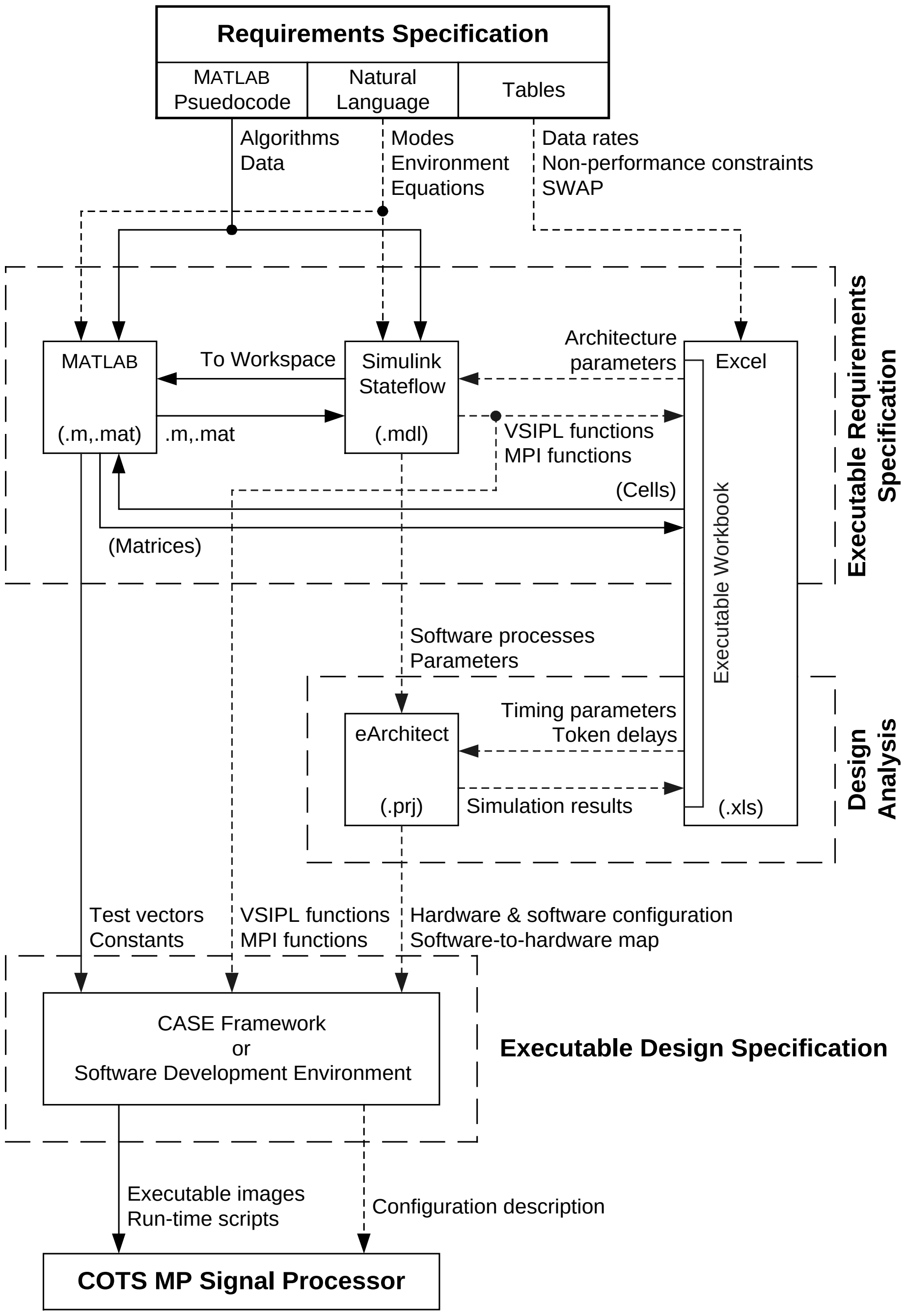

Fig. 3. MAGIC SDM information flow and illustration of model continuity.

VSIPL is an API supporting portability for COTS users of real-time embedded multicomputers that has been produced by a national forum of government, academia, and industry participants [3]. VSIPL is computational middleware, which also supports interoperability with interprocessor communication (IPC) middleware such as MPI and MPI/RT. The VSIPL Forum has produced the API, a prototype reference library, and a test suite to verify API compliance. Commercial implementations are just now becoming available (early 2000). Earnest consideration by various defense programs as well as other commercial projects is underway and early adoption has begun. The VSIPL API standard provides hundreds of functions to the application software developer to support computation on scalars, vectors, or dense rectangular arrays.

Canonical development of embedded signal processing applications using COTS multiprocessing hardware and software typically consists of partitioning the code into two portions. One portion is the "outer loop" where the setup and cleanup functions are executed, typically memory allocation and coefficient generation, such as FFT twiddle factors and window coefficients. The other portion is the "inner loop" where the time-critical repetitive streaming data transformation functions lie. A VSIPL application will be built similarly, with the outer loop executing heavyweight system functions that allocate memory when creating blocks and parameterized accessors called views. The block creation is substantial, while the view object handles take up very little memory, but do require system support.

Message passing is a powerful and very general method of expressing parallelism and can be used to create extremely efficient parallel software applications. It has become the most widely used method of programming many types of parallel computers. High-performance implementations of MPI are now available, including implementations for COTS MP platforms. The leading vendor is MPI Software Technology, Inc. (MSTI) who provides high-performance implementations of MPI under the commercial trademark MPI/PRO for NOWs and SPCs, including two of the three leading COTS MP vendors in our technology space (RACEway and Myrinet). There is another standards effort underway to specify a real-time version of MPI with a guaranteed quality-of-service (QoS) called MPI/RT [4]. Non-QoS beta versions of MPI/RT are just now (early 2000) beginning to appear.

5 Using VSIPL & MPI for Model Continuity

The two most important reasons for choosing VSIPL and MPI are acceptable performance and that they were standards-based. If these middleware could not deliver performance commensurate with the vendors' native computational and communications libraries, they would not be as useful and therefore less acceptable. However, preliminary VSIPL benchmarks recently released by one COTS MP vendor (Mercury Computer Systems) shows computational throughput achieving up toward 98% of their native algorithm library. MPI benchmarks released by one commercial MPI vendor (MSTI) show bandwidths within 5% of the RACE theoretical maximum for large block sizes, which is very close to that achieved by the vendor’s own native communication library.

Being standards-based is the other key characteristic of these middleware. The participation of researchers, implementers, and users to form and support these standards goes a long way towards assuring their adoption. It is our opinion that there are two types of standards, official and de facto. Being standard is not a blessing deferred by some official “acronym’d” organization, but something established de facto when companies invest their own resources in products designed to a standard and consumers purchase those products. We are not saying that oversight and management by standards organizations is not worthwhile, we are just saying that real standards are determined by the community. Suffice to say, MPI and VSIPL are currently establishing themselves in the marketplace as standards, and no doubt “official sanctification” will occur sometime later.

Being a genuine de facto standard means that code generated within the MAGIC SDM can be used to estimate communication and computation token delays in performance modeling, as well as for the inner-loop computational code in the implementation. This strengthens the thread of continuity from specification to design (token delays) and implementation (inner-loop code).

6 Conclusion

We have introduced a new specification and design methodology (SDM) in this paper, the MAGIC SDM, that leverages standards-based middleware to achieve model continuity in the specification and design of signal processing systems implemented with COTS hardware and software. This is feasible since middleware generated in the specification and design processes can be used in the physical implementation because of the efficiency of both the VSIPL computation and MPI communication middleware.

References

[1] R. S. Janka and L. M. Wills, “Virtual Benchmarking of Embedded Multiprocessor Signal Processing Systems,” in *submitted to IEEE Design and Test of Computers*, 2000, pp. 26.

[2] R. S. Janka, “A Model-Continuous Specification and Design Methodology for Embedded Multiprocessor Signal Processing Systems,” a Ph.D. dissertation in the School of Electrical and Computer Engineering. Atlanta, Georgia: Georgia Institute of Technology, 1999, pp. xxiii, 225.

[3] VSIPL Forum, “VSIPL v1.0 API Standard Specification,” DARPA and the Navy, Draft http://www.vsipl.org/PubInfo/pubdrftrev.html, 1999.

[4] Real-Time Message Passing Interface (MPI/RT) Forum, “Document for the Real-Time Message Passing Interface (MPI/RT-1.0) Draft Standard,” DARPA, Draft http://www.mpirt.org/drafts.html, February 1, 1999.