

Combining Virtual Benchmarking with Rapid System Prototyping For Real-Time Embedded Multiprocessor Signal Processing System Codesign

Randall S. Janka
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332-0856 USA
randall.janka@gtri.gatech.edu

Linda M. Wills
School of Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250 USA
linda.wills@ee.gatech.edu

Abstract

The codesign of embedded real-time signal processing systems is complex. The use of commercial-off-the-shelf (COTS) multiprocessor (MP) hardware and software can reduce codesign complexity. Further complexity reduction can be obtained with emerging rapid system prototyping (RSP) frameworks, which can generate deployable code by leveraging vendor communication and computation libraries. However, these RSP frameworks are inadequate in providing a sound specification and design methodology (SDM) because they require the designer to first choose the implementation target before specification and design exploration. We have developed a new SDM known as MAGIC that allows the designer to capture the specification in an executable model that can then be used in design exploration to find the optimal COTS MP technology and architecture before committing to that technology. The MAGIC SDM exploits emerging open-standards based VSIPL computation middleware and MPI communication middleware to provide connectivity between specification and design with RSP frameworks for implementation.

1 Introduction

The process of designing large real-time embedded signal processing systems is plagued by a lack of coherent specification and design methodology (SDM). A canonical waterfall design process is commonly used to specify, design, and implement these systems with commercial-off-the-shelf (COTS) multiprocessing (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. COTS MP Technology. 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, thus making the current incoherent methodology costly. This paper shows how designs targeting COTS MP technologies can be improved by providing a coherent coupling between these frameworks, a quality known as “model continuity.”

The codesign of embedded real-time signal processing systems is complex and made more difficult when pressed by time-to-market. The use of COTS MP hardware and software is usually required by the customer and/or necessitated by time-to-market. This can reduce the complexity of the codesign by constraining the hardware search space. Further complexity reduction can be obtained by the use of emerging rapid system prototyping (RSP) frameworks created for COTS MP-based codesign. These frameworks provide GUI canvases for hardware and software design as well as configuration and integration, and also possess effective code generation that produces deployable code by leveraging vendor communication and computation libraries.

While these RSP frameworks are great aids for implementation, they are inadequate in providing a sound specification and design methodology (SDM) because they require the designer to first choose the implementation target *before* specification and design exploration. We have developed a new SDM known as MAGIC that allows the designer to capture the specification in an executable model that can then be used in design exploration to find the optimal COTS MP technology *before* committing to that technology. The design exploration uses “virtual benchmarking” based on performance modeling to determine the optimum configuration for a given technology. The RSP frameworks can then consume the design model to generate deployable code, configuration files, and run-time scripts.

The ability to pass an executable model between the phases of specification, design, and implementation is a quality known as model continuity. The MAGIC SDM possesses strong model continuity enabled in large measure by

emerging open-standard middleware: VSIPL computation middleware and MPI communication middleware.

2 Application and Technology Domain

The application space addressed by our methodology and the technologies under consideration is real-time embedded signal processing. This includes radar and sonar signal processing, medical imaging, and other applications characterized by high-bandwidth signal input that is on the order of tens to hundreds of megabytes per second and high-throughput computation requirements on the order of tens to hundreds of gigaflops/second. After system initialization, these systems usually have a single data transformational state, but some systems are modal, requiring multiple reactive states.

The use of COTS technology in our application domain means using a PCI or VME chassis containing multiprocessor boards with high-speed interprocessor bandwidth and C language support. Table 1 shows the leading technologies used for implementing these systems. Despite limiting the design space to a finite number of hardware options, the design process has still been challenging, given compressing development cycles that increase software development productivity requirements, implicitly requiring that software be portable, so that previous design and development efforts can be reused. This productivity and portability must be achievable without an appreciable loss of system performance.

Table 1. Technologies in our application space.

Vendor	Interconnection	Processors Supported
CSPI	Myrinet	i860, SHARC, PPC
Mercury Computer Systems	RACE RACE++	i860, SHARC, PPC & Altivec
SKY Computers	SKYchannel	i860, SHARC PPC & Altivec

2.1 RSP Frameworks

A partial response to this design challenge of real-time multiprocessor digital signal processing systems has been the development of different graphically-based frameworks of tools, such as GEDAE¹, RIPPEN², PGM ACT³, and

¹ GEDAE–Graphical Entry, Distributed Application Environment from Lockheed Martin ATL.

² RIPPEN–Real-time Interactive Programming and Processing Environment from ORINCON.

PeakWare for RACE⁴ [1, 2] to provide computer-aided system engineering (CASE) support for system implementation. In particular, these frameworks offer code generation that reduces the complexity of system configuration and communication coding, a quality known as “complexity control.” Yet no one single framework or one single language can cover the entire design process. Powerful implementation tools such as these RSP frameworks can generate deployable application code, but are weak in capturing requirements and are difficult to use in exploring architecture design alternatives. Some languages, such as MATLAB, are powerful in capturing computational requirements, but do not readily lend themselves to being used for deployed implementations.

2.2 Ideal COTS MP SDM Flow

Ideally, the information flow for specification and design would occur as shown in Figure 1. In this ideal SDM, the executable specification model is passed into the design analysis phase, and the design model in the form of an executable design specification is then passed into the implementation phase, where the physical implementation occurs.

2.3 Current COTS MP SDM Flow

Unfortunately, the current state of practice in this domain does not support model continuity, which is illustrated in Figure 2. While the different frameworks noted in §2.1 have evolved to support the design and rapid system prototyping of signal processing systems using COTS MP technology, they are fundamentally sophisticated software development environments (SDEs) providing a powerful framework for the codesign of a COTS MP-based system. Codesign in this application and technology domain consists of the following:

- Board-level hardware description capture
- Software process and function description capture
- Software-to-hardware mapping
- Generation of code, makefiles, and run-time scripts

Consequently, despite having powerful frameworks to support codesign, specification and design still suffers from model *discontinuity* as illustrated in Figure 2.

³ PGM ACT–Autocoding Toolset (ACT) using the Processing Graph Method (PGM).

⁴ PeakWare for RACE–Codesign framework from Mercury Computer Systems that is layered on “Talaris,” their configuration middleware.

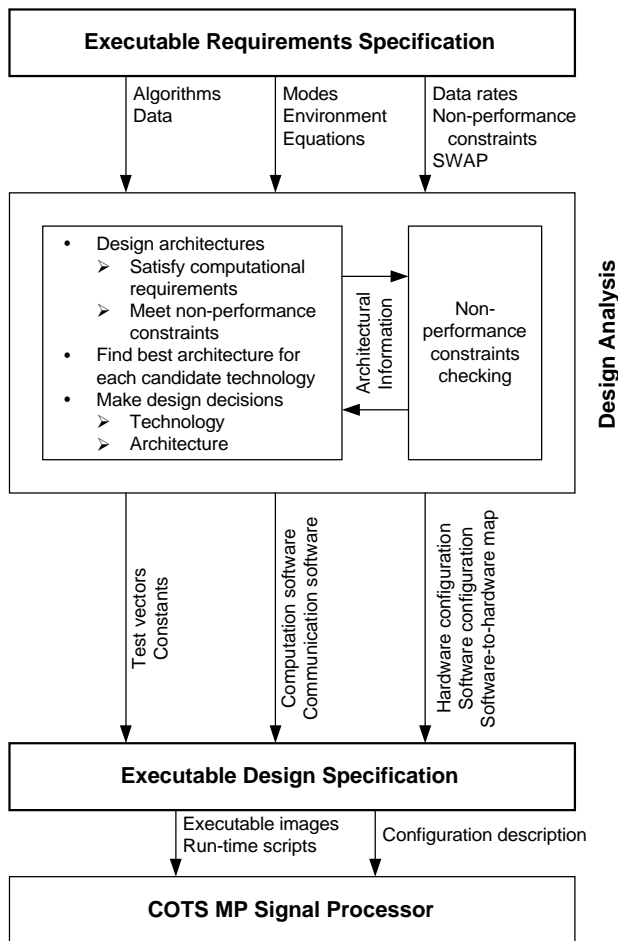


Figure 1. Basic flow of information needed to support model continuity.

3 The MAGIC SDM

The new SDM that we have developed is the MAGIC SDM, which stands for the “methodology applying generation, integration, and continuity.” [3] The origin of this moniker will become clear as we discuss how the MAGIC SDM is used.

Any SDM will start with some human language text requirements specification document. The goal of SDMs is to go from this inexact document to a design and implementation in a manner that minimizes propagation of specification and/or design errors. We do this with an integration of tools guided by sound rules to capture the requirements in a format to make sure there are no conflicts or absence of requirements, then proceed on through a vendor-independent design phase. Without first committing to a vendor, alternate architectures can be considered and an optimum one decided upon. We then take code generated from specifica-

tion and software-to-hardware mapping determined from design to provide inputs to an implementation framework.

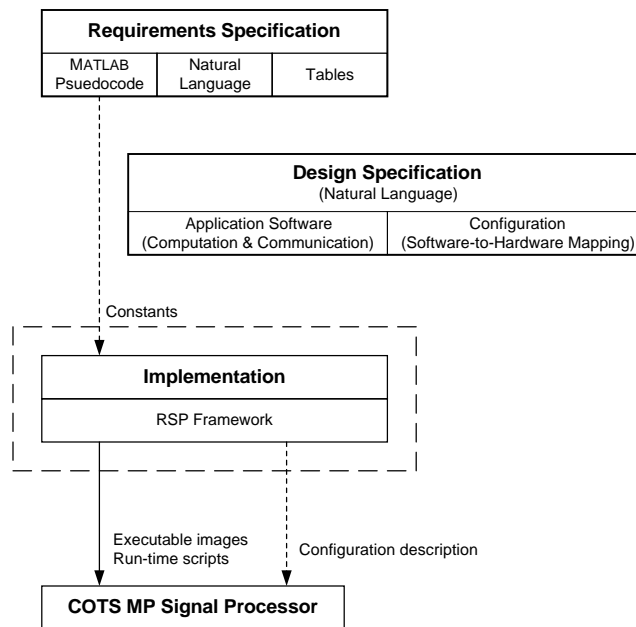


Figure 2. How model continuity is currently lacking in current COTS MP SDM.

The starting point for specification and design in our domain is the set of computation requirements. These are algorithms and data “specified” by MATLAB code, including different scenarios of inputs and their associated outputs. The MATLAB code serves well as an input to a framework that can use it to create an executable specification. The scenarios will provide valuable inputs for the generation of test data to be used downstream in the implementation phase. Communication and control requirements typically refer to data I/O rates as well as the signal processor modes and the control signals that determine the processor’s mode (state). Processors in our domain have few states; often there are two: one state for initialization and setup (“outer loop”) and one state for steady state data transformation (“inner loop”). These modes must be defined and described, preferably in an executable model. Constraints include SWAP, latencies, reliability, and other “illities,” which are usually tabulated. It would be useful to have these data encapsulated in a fashion that allows us to include their verification during the specification and design iterations. We now redraw the ideal SDM of Figure 1 as our new SDM, shown in a simplified diagram of the specification and design flow in Figure 3.

A specification and design methodology is comprised of “tools and rules,” so we now lay out the frameworks (§3.1) and middleware (§3.2) used to implement the MAGIC SDM, and then delineate the rules (§3.3).

3.1 MAGIC SDM Tools

We have chosen the following frameworks to integrate into the MAGIC SDM. We did not choose them because they are perfect; almost all frameworks targeting complex systems are more accurately described not as “frameworks” but as “frameworks-in-progress.” We have chosen the frameworks described in this section because they are well-matched to our application and technology domain, as well as stable commercial products.

For requirements capture and modeling we have chosen the DSP Workstation (DSPW) from The MathWorks as well as Excel and Excel Link (also from The MathWorks). For design exploration we chose eArchitect from Viewlogic, which provides a performance modeling framework with the necessary COTS MP component models. Characterization and features driving the selection of The MathWorks and Viewlogic frameworks are given in the following paragraphs where we discuss these frameworks.

3.1.1 DSP Workstation

MATLAB is the de facto lingua franca of algorithm developers, including radar signal processing system analysts. Simulink is a system modeling framework strongly tied to MATLAB, allowing MATLAB expressions to be used explicitly in Simulink blocks. Supplemented by the DSP Blockset, Simulink has become a viable rapid prototyping environment for DSP applications. The Real-Time Workshop (RTW) is the C code generation facility complementing Simulink. It is especially useful because The MathWorks is moving towards adding VSIPL computational middleware support to RTW, an effort we are supporting.

3.1.2 Excel and Matlab Excel Link

Excel provides the framework needed for requirements tabulation and analysis. The Excel spreadsheet is a commodity productivity application familiar to all, and in the same way that many analysis and design frameworks provide support for MATLAB, tabular data-oriented frameworks provide support for Excel. Excel Link is a facility rather than a framework. It is a channel to allow Excel to copy data into MATLAB and execute on it in MATLAB while remaining in Excel. Many requirements are easily “captured” in a spreadsheet, and depending on the sophistication of the computation required to iterate between requirements modeling and design analysis, Excel or MATLAB may be required. Excel Link allows the specifier to remain in a single framework.

3.1.3 eArchitect

Performance modeling was chosen for design exploration and analysis since it supports architectural trade-off

analysis without prematurely committing to a given vendor’s hardware and software. The COTS performance modeling framework that is best matched to our domain will provide support for the technologies most likely to be used for implementing the signal processor. There are few embedded multiprocessing performance modeling frameworks available commercially. We are only aware of one that supports VME and at least two of the COTS MP interconnection technologies (RACEway and Myrinet), and that is eArchitect from Viewlogic.

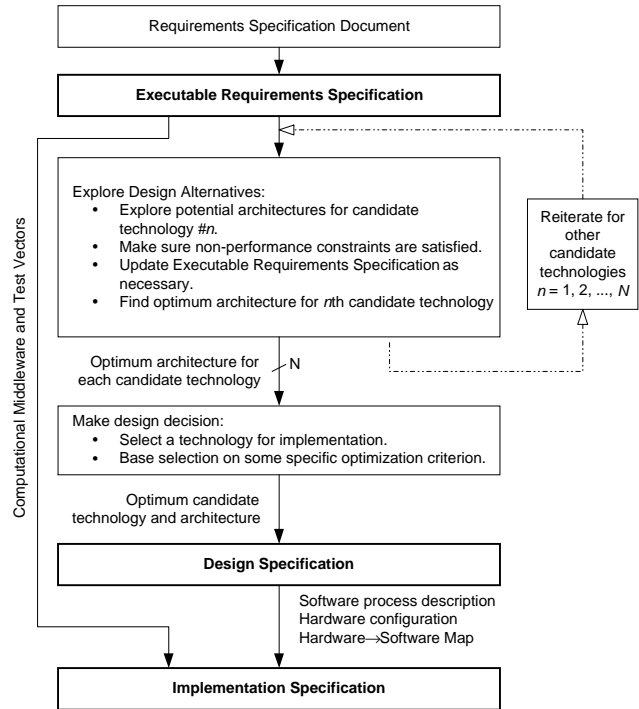


Figure 3. Simplified view of new MAGIC SDM.

3.2 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.

Simulink's Real-Time Workshop generates middleware for computation using VSIPL, MPI for communication, and/or MPI/RT for communication and control produces code for both design and implementation. The generated middleware is then 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.

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 [4]. VSIPL is computational middleware, which also supports interoperability with interprocessor communication (IPC) middleware such as MPI and MPI/RT. 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.

Message passing is a powerful and very general method of expressing parallelism and can be used to create extremely efficient parallel applications. 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, 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 [5]. Non-QoS beta versions of MPI/RT are just now (early 2000) beginning to appear.

To generate the steady-state inner-loop middleware-based C code from Simulink, the DSP Blockset is translated

into VSIPL or MPI function calls with the arguments determined by the parameters contained in the Simulink blocks. Basically, Simulink "boxes" are transformed into VSIPL computation function calls, while the "arrows" are transformed into MPI communication function calls.

3.3 MAGIC SDM Rules

We lay out here the specification and design rules of the MAGIC SDM. We assume that a natural language (e.g., English) requirements specification document exists that contains the system requirements, interfaces, data rates, etc. We do not assume that the algorithms have been coded in MATLAB, though it would be very unusual for them not to be.

- 1) Tabulate requirements—Identify and cull details of the requirements from the requirements specification document.
- 2) Capture non-constraint requirements in an executable model—Describe computation, communication, and control requirements in an executable model.
- 3) Build executable workbook with requirements—Put all the requirements into a tabular form to facilitate computational manipulation, e.g., in a worksheet/workbook environment such as Excel.
- 4) Gather benchmarks for tokens—Gather benchmarks of the middleware functions that are likely to be used in design and implementation and enter them into the executable workbook.
- 5) Explore alternative architectures and technologies—Use performance modeling to explore potential architectures for a given technology, then determine the best architecture for that technology. Repeat as necessary for other candidate technologies.
- 6) Make design decisions—Decide which technology and architecture to use in implementing the signal processor.
- 7) Create implementation specification—Pass along architectural details to the system implementation specification based on the design exploration.

4 Case Study

We have demonstrated the use of the MAGIC SDM with the use of a real-world domain-relevant benchmark, the RASSP⁵ [6] SAR benchmark. We have also shown that the

⁵ RASSP—Rapid Prototyping of Application Specific Signal Processors; a DARPA program exploring design methodologies in this application and technology domain.

MAGIC SDM accomplishes three important goals as discussed in the following paragraphs.

4.1 The MAGIC SDM works as postulated, which means the rules can be followed and the tools work—especially in providing model continuity.

Examples of model continuity included the passing of requirements model information back and forth to our design analysis performance modeling via our executable workbook, which also assured non-performance constraints were satisfied. Also, once a design was chosen, our requirements model was used to generate inner-loop computation and communication C code as well as test vectors that can all be used in the processor's implementation. The performance model provided hardware configuration and software-to-hardware mapping information to the implementation.

4.2 The MAGIC SDM yields benchmarks of a full frame of data with run-times beyond the 3-second latency requirement, which is 20 times the longest VHDL simulation.

We have run the VHDL-based virtual benchmarks anywhere from 1.5 seconds to 4.0 seconds, well over the 150 ms achieved in other RASSP SAR VP-based VHDL simulations [6].

4.3 The MAGIC SDM works in providing the framework to evaluate competitive technologies prior to implementation, which the CASE SDMs cannot do at all.

We have demonstrated this by examining different architectures using real processor computation and communication deterministic benchmarks used to perform system performance simulations that include the nondeterministic interprocessor communication. We were able to determine which architectures would satisfy performance and non-performance requirements, then decide on the optimum architecture for a given technology. This enabled us to specify the implementation, including its hardware configuration, software processes, software-to-hardware mapping, inner-loop implementation code, and test vectors for implementation verification.

5 Conclusion

It is important to consider one fundamental issue, which is to determine how applicable the MAGIC SDM is beyond the domain of real-time streaming data with data transfor-

mation processing applications implemented with embedded COTS multiprocessing technology. By constraining our focus to our application domain, we were therefore able to identify the frameworks and middleware that would be viable for integration into the MAGIC SDM. While we have constrained our focus to this domain, it seems promising to adapt the MAGIC SDM to other application and technology domains. Frameworks exist for other application and technology domains, usually referred to as "EDA" (electronic design automation). Middleware is by no means restricted solely to our technology domain, and could similarly serve as the model continuity medium in other technology domains. Using the right tools at the right time should be applicable to other domains as well. We believe this would be worthwhile to pursue, especially in the system-on-a-chip domain.

References

- [1] R. Janka, "A New Development Framework Based on Efficient Middleware for Real-Time Embedded Heterogeneous Multicomputers," presented at 1999 IEEE Conference and Workshop on Engineering of Computer-Based Systems (ECBS '99), Nashville, Tennessee, 1999.
- [2] R. Janka, "Models of Computation for Specification and Design Methodology Frameworks for Parallel and Distributed Real-Time Embedded Multiprocessor Signal Processing Systems," presented at The 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), Las Vegas, NV, 1999.
- [3] 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.
- [4] VSIPL Forum, "VSIPL v1.0 API Standard Specification," DARPA and the Navy, Draft <http://www.vsipl.org/PubInfo/pubdrftrev.html>, 1999.
- [5] 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.
- [6] M. A. Richards, A. J. Gadiant, and G. A. Frank, "Rapid Prototyping of Application Specific Signal Processors," in *Journal of VLSI Signal Processing*, vol. 15, S. Y. Kung, Ed., 1 ed. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1997, pp. 200.