Multi-language System Design

Ahmed Jerraya
TIMA Laboratory
Grenoble, France
Ahmed.Jerraya@imag.fr

Rolf Ernst
Technical University Braunschweig
Braunschweig, Germany
ernst@ida.ing.tu-bs.de

Abstract

The design of large systems, like a mobile telecommunication terminal or the electronic parts of an airplane or a car, may require the participation of several groups belonging to different companies and using different design methods, languages and tools. The concept of multi-language specification aims at coordinating different cultures through the unification of the languages, formalism, and notations.

This hot topic discusses the main issues and approaches to multi-language design. Two research directions are currently explored by the EDA community. The first is based on the computation models underlying the languages while the second deals with the specification languages themselves.

1. Multi-language design

Most existing system specification languages are based on a single paradigm. Each of these languages is more efficient for a given application domain. For instance some of these languages are more adapted to the specification of state-based specification (SDL or Statechart), some others are more suited for data computation (LUSTRE, SILAGE), while many others are more suitable for algorithmic description (C, C++).

When a large system has to be designed by separate groups, they may have different cultures and expertise with different modeling styles. The specification of such large designs may lead each group to use a different language which is more suitable for the specification of the subsystem they are designing according to its application domain and to their culture.

Figure 1 shows a typical complex system, a mobile telecommunication terminal, e.g. a G.S.M. handset. This system is made of four heterogeneous subsystems that are traditionally designed by separate groups that may be geographically distributed.

a) The protocol and MMI subsystem :
This part is in charge of high-level protocols and data processing and user interface. It is generally designed by a software group using high-level languages such as SDL or C++.

b) The DSP subsystem :
This part is in charge of signal processing and error correction. It is generally designed by "DSP Group using specific tools and methods such as Matlab/Simulink [1] or COSSAP [2].

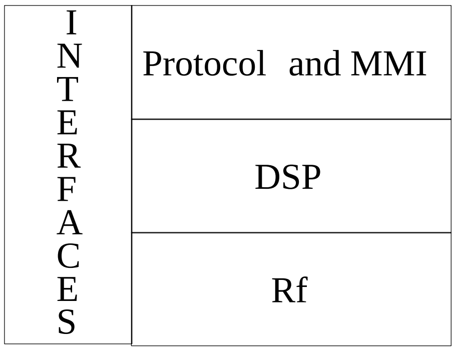

Figure 1: Heterogeneous Architecture of a Mobile Telecom Terminal e.g. a G.S.M. handset

c) the DSP subsystem :
This part is in charge of the physical connection. It is generally made by an analog design group using another kind of specific tools and method such as CMS [3].

d) The interface subsystem :
This part is in charge of the communication between the three other parts. It may include complex buses and a sophisticated memory system. It is generally designed by a hardware group using classical EDA tools.

The key issue for the design of such a system is the validation of the overall design and the synthesis of the interfaces between the different subsystems. Of course, most of these subsystems may include both software and hardware.

Figure 2 shows a generic flow for co-design starting from multi-language specification. Each subsystem of the initial specification is described in a specific language and may need to be decomposed into hardware and software parts. Moreover, we may need to compose some of these subsystems in order to perform global hardware/software partitioning. In other words, partitioning may be local to a given subsystem or global to several subsystems. The co-design process also needs to tackle the refinement of interfaces and communication between subsystems.

The problems of interfacing and multi-language validation need to be solved. In addition, this model brings about another difficult issue: language composition. In fact, in the case where a global partitioning is needed, the different subsystems need to be mapped onto a homogeneous model in order to be decomposed. This operation would need a composition format able to accommodate the concepts used for the specification of the different subsystems and their interconnection.

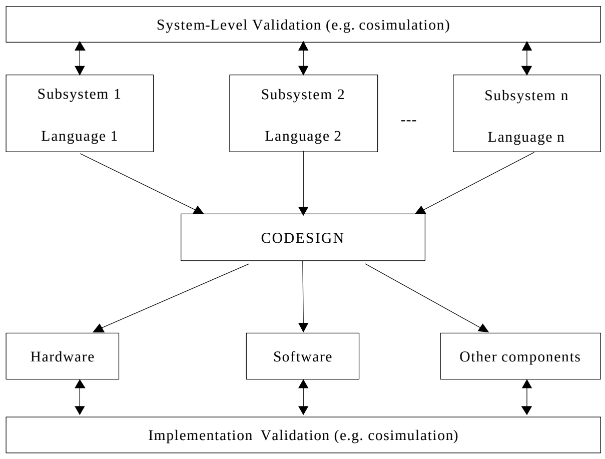

Figure 2: Multi-language codesign

The research community is currently exploring two directions to solve the multi-language design problem. The first direction makes use of the computation models underlying the languages. In this case formal methods are developed for reasoning about multi-language specification and design. The second direction deals with the specification languages themselves. These directions are presented in the next two sections

2. Computation models-based approaches

The goal is to focus on the computation models underlying specification languages and architectures. The key message delivered here is that despite the proliferation of specifications there are only a few basic concepts and models underlying all these languages.

In fact, most co-design tools start by translating their input language into an intermediate form that corresponds to a computation model which is easier to transform and to refine. Specification languages differ mainly in their approach to provide a view of the basic components, the link between these components and the composition of components. Basic components are described as behavior, these may be reactive or transformative. The links fix the inter-module communication and the composition fixes the hierarchy.

Many embedded system applications consist of a combination of reactive and transformative functions. Often, several languages with different underlying models of computation are used for the specification and modeling of different functions of an individual system. The languages are selected because of their particular suitability for certain applications and optimizations, or because they have become generally accepted as a standard within an application domain. Traditionally, flow-oriented models of computation are used in the area of signal processing or control engineering. In the last two decades, these were enhanced by models which are more suitable for the analysis of the design space and thus for synthesis, from synchronous data flow (SDF) to boolean data flow to dynamic data flow. A very instructive comparison of such representations can be found in [4]. These models are already accepted in industrial design, driven by an extensive set of design tools which also support application optimization, as for instance MatrixX (Integrated Systems) or MATLAB/SIMULINK (Math Works). A similar development can be observed for event-oriented models of computation, which are of special importance for reactive systems, e. g. STATEMATE in automotive engineering or SDL in the area of telecommunications. The lack of coherency of the different languages, methods and tools is a substantial obstacle in design space exploration and system optimization [5]. Language interfaces and library standardization or tool efficiency improvements alone will not be sufficient for improved system optimization as we will explain in the following.

Design space exploration and system optimization require the knowledge of resource utilization and timing. Relevant resources are processors (or hardware), communication channels and memory. Timing is required for scheduling. Accuracy and completeness of this knowledge decide on the possible degree of optimization. Inaccuracies and incompleteness result from abstraction, from unknown or from data dependent behavior. In the context of design with multiple languages, different abstractions are the main source for the lack of accuracy and completeness. Unknown timing or resource utilization is, in general, due to limited analysis capabilities (practical or theoretical limitations) and becomes a particular problem when third-party software or „legacy code“ must be included.

Languages differ in the timing and resource utilization data which they provide. Flow graphs explicitly model the amount of data which is produced and consumed in each execution. This simplifies scheduling and resource allocation, especially when timing constraints are restricted to input and output sample rates. If data consumption and production are time or state dependent, as in boolean data flow graphs, then the sequence of executions must be modeled for scheduling and resource allocation, as well, or worst case behavior must be assumed leading to less efficient solutions. In contrast, programming languages, such as C, permit almost arbitrary fixed or data dependent communication which is difficult to analyze. Such C programs are typically invoked periodically or in response to input events rather than activated by the availability of input data tokens. A third example are SDL processes which expose the data flow, but communication is state dependent and processes are activated by input data. Combining such different execution models in a single scheduling and resource allocation strategy with verifiable worst case behavior which exploits the specific information provided by the different models is a challenging task. There are basically two approaches. The first one is the definition of a single model of computation with a single unifying semantics which covers a variety of models and the second one is the use of a simpler model with behavior uncertainty intervals which are just sufficient to model all information required for scheduling and resource allocation, i.e. for target system optimization. The presentations review some of the recent trends which are, e.g. represented by [6,7] or [8].

3. Language-based approaches

The second approach to multi-language design is based on the languages themselves. The key issues with such a scheme are validation and interfacing. The use of multi-language specification requires new validation techniques able to handle a multi-paradigm model. Instead of simulation we will need co-simulation and instead of verification we will need co-verification. Additionally, multi-language specification brings about the issue of interfacing subsystems which are described in different languages. These interfaces need to be refined when the initial specification is mapped onto a prototype. In fact, the global configuration of the system is a kind of "system-level netlist" that specifies the interconnection between different subsystems. Since different languages may be based on different concepts for data exchange, the interpretation of the link between subsystems is generally a difficult task.

There are two main approaches for multi-language design: the compositional approach and the co-simulation-based approach.

The compositional approach (cf. Figure 3) aims at integrating the partial specification of sub-systems into a unified representation which is used for the verification and design of the global behavior. This allows to operate full coherence and consistency checking, to identify requirements for traceability links.

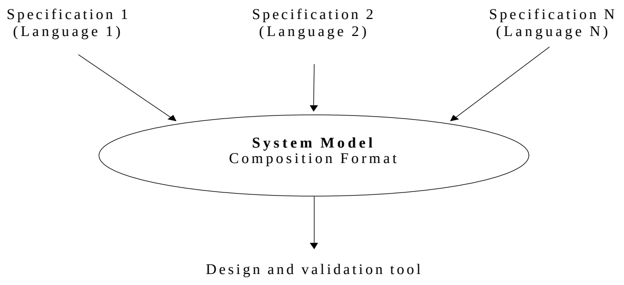
Specification 1 (Language 1)
Specification 2 (Language 2)
Specification N (Language N)
System Model
Composition Format
Design and validation tool

Figure 3: Composition-based multi-language design

Polis, Javatime and SpecC detailed respectively in [9, 10, 11] introduce a compositional-based codesign approach. Polis uses an internal model called Co-design FSMs for composition. Both Javatime and SpecC use another specification language (Java and SpecC) for composition.

The cosimulation based approach (cf. Figure 4) consists in interconnecting the design environments associated to each of the partial specification. Compared with the deep specification integration accomplished by the compositional approaches, cosimulation is an engineering solution to multi-language design that performs just a shallow integration of the partial specifications.

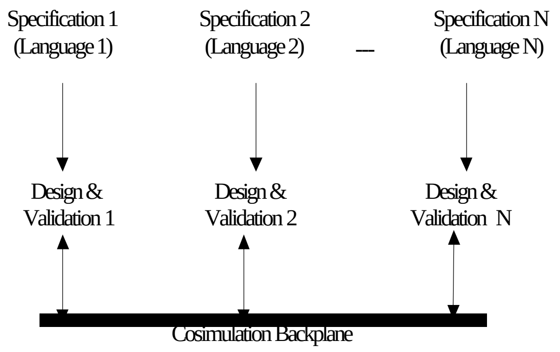

Figure 4: Co-simulation based multi-language design

The cosimulation-based approach is emerging in the EDA industry with the availability of hardware/software cosimulation [12, 13, 14]. With these methods hardware parts are described in VHDL or Verilog and the software part is described in C or C++. The configuration of the overall system is generally described using an ad-hoc language in order to specify the interconnection between the hardware blocs and the software block. Most existing cosimulation methods solve the multi-language design problem at the RTL-level. However, a new generation of multi-language approaches is trying to raise the abstraction level of the design [15, 16]. The goal is to be able to start from system-specification languages such as SDL, COSSAP or Matlab/Simulink.

1. MATLAB® 5 / SIMULINK® 2 : Mathworks Inc. - http://www.mathworks.com
2. Synopsys. 1997 (Jan.). COSSAP (Reference Manuals). Synopsys
3. HP. 1998. HP Advanced Design System.[ADS] http://www.tmo.hp.com/tmo/hpeesof/products/ads/adsoview.html
4. E. A. Lee, Th. M. Parks. Data Flow Process Networks. Proceedings of the IEEE, vol. 83, no. 5, 95, pp. 773-799.
5. R. Ernst. Codesign of Embedded Systems: Status and Trends. IEEE Design & Test, vol. 15, no. 2, 98, pp. 45-54.
6. T. Grötker, R. Schoenen, H. Meyr. PCC: A Modeling Technique for Mixed Control/Data Flow Systems. Proc. ED&TC 97, pp. 482-486.
7. E. A. Lee. Modeling Concurrent Real-Time Processes using Discrete Events. Annals of Software Engineering, 98.
8. D. Ziegenbein, K. Richter. R. Ernst, J. Teich, L. Thiele. Combining Multiple Models of Computation for Scheduling and Allocation. Proc. 6th Int. Workshop on Hardware/Software Co-design CODES, Seattle, March 98, pp. 9-13.
9. F. Balavin et al. "HW/SW Codesign" The POLIS approach. Kluwer Academic Publishers, 1997.
10. D. Gajski, R. Dömer and J. Zhu. IP-Centric Methodology and Design with SpecC Language, chapter in "System-level Synthesis", NATO ASI 1998 edited by A. Jerraya and J. Mermet, Kluwer Academic Publishers, 1999.
11. J. Shin et al., The Javatime Approach to Mixed HW/SW System Design, chapter in "System-level Synthesis", NATO ASI 1998 edited by A. Jerraya and J. Mermet, Kluwer Academic Publishers, 1999
12. K.Van Rompaey, D. Verkest, I. Bolsens, and H. De Man. Coware - a design environment for heteregeneous hardware/software systems. In Proceedings of the European Design Automation Conference. Geneve, September 1996.
13. C.A. Valderrama, A. Changuel, P.V. Raghavan, M. Abid, T. Ben Ismail, and A.A. Jerraya. A unified model for co-simulation and co-synthesis of mixed hardware/software systems. In Proc. European Design and Test Conference (EDAC-ETC-EuroASIC). IEEE CS Press, March 1995.
14. W.M. Loucks, B.J. Doray, and D.G. Agnew. Experiences in real time hardware-software cosimulation. In Proc. VHDL Int'l Users Forum (VIUF), pages 47--57, April 1993.
15. A. Jerraya et al., Multilanguage Specification for System Design, chapter in "System-level Synthesis", NATO ASI 1998 edited by A. Jerraya and J. Mermet, Kluwer Academic Publishers, 1999
16. Ph. Le Marrec et al. HW/SW and mechanical cosimulation for Automotive Applications. RSP'98, Belgium, 1998.