FRAMEWORK FOR SYSTEM DESIGN, VALIDATION AND FAST PROTOTYPING OF MULTIPROCESSOR SYSTEM–ON-CHIP:

Applied to Telecommunication Systems

N.E.Zergainoh, A.Baghdadi, L.Tambour, D.Lyonnard, A.A.Jerraya

1. Introduction

Designers of today's telecommunication products such as cellular phones, wireless, and networking devices are facing a rapidly growing system complexity. Driven by the advances in semiconductor technology and the need for new telecommunication applications, the amount of functionality that is realized on a SOC is increasing enormously. The architecture for these applications are truly heterogeneous multiprocessor including hardware/software, and digital/analog parts.

The growing complexity of these systems requires that design tools and methods work at increasingly higher levels of abstraction. Abstraction helps manage complexity because systems can be specified by their behavior instance of their structure. New methods are needed to handle the design and validation of these systems where different languages, tools and models need to be used for full system specification, simulation and design. Systems must be described and specified formally if they are to be analyzed or manipulated by other design tools. Simulation, i.e., execution of specification, helps users validate a specification with respect to both functionality and properties. Designs can be synthesized and transformed from abstract specifications into physical implementations. Verification tools analyzed designs and requirements and provide assurance that designs have been properly implemented and that they will always work correctly.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35409-5_23

B. Kleinjohann (ed.), *Architecture and Design of Distributed Embedded Systems*

This chapter presents a methodology and framework for system design, validation and fast prototyping of advanced telecommunication multiprocessor System–On-Chip. The presented methodology takes into account multi-languages functional specification, multi-level validation, both algorithm and multiprocessor architecture explorations, refinement, software and hardware targeting, and final prototyping generation. The framework combines a different technologies such as Matlab™, SDL and codesign tool, as well as other related tools, to support all development cycle. A case study on real world communication device application demonstrates the effectiveness of the methodology and how the framework, was applied to this real industrial product to increase the productivity and the quality of the SOC design effort.

The remainder of this chapter is organized as follows. Section 2 gives an overview of our global system-level design methodology. Section 3 presents the framework for system validation and prototyping of digital systems. Section 4 describes the specification and design of 2.4 kbits/s LPC vocoder and highlights the results obtained. Section 5 discusses the results and presents the lessons learned. Section summarizes the chapter.

2. Overview of a generic methodology

The methodology starts from a high-level specification describing the functionality of the each subsystem as a hierarchical mixed data/control flow diagram. Note that description already can contain components from a reuse library. The design process starts with the designer creating a functional specification of each subsystem. The aim is to validate and explore the algorithms and functionality of system by system-level simulation (i.e. co-simulation) including the validation of the individual subsystem, and the full system. The various algorithms may be explored through simulation. Therefore, once the system functionality and algorithms are validated by system-level simulation (i.e. co-simulation), we use, in addition, a back-annotation approach. At this stage we obtain system-level specification associated to the back-annotate performance models. The aim is to explore different architectures through the system-level simulation. Thanks to this new time-annotated specification, it is possible to predict the performance of all feasible architecture solutions with a good trade-off between speed and accuracy. Then once an architecture is decided, the functional specification is mapped into an architectural specification. The digital part of the system is partitioned into hardware and software components. Other subsystems also may be refined. The architectural specification, may include one or more processors and others components. At this stage we obtain a multi-processors architecture without technology. The aim is to validate through the fast co-simulation, thanks to the abstract model of architecture. Once an architecture is validated, the targeting will proceed from the top level SoC architecture and the individual components contained on the SoC. In this stage, the architectural model is more detailed and the cycle-true validation is obtained through slow co-simulation.

3. Framework for system validation and prototyping

In order to support all development cycle of telecommunication systems, the framework combines a different technologies such Matlab and SDL languages, co-design tools, performance estimator and co-simulation environment. The framework start from Matlab and produces a heterogeneous multiprocessor system-on-chip. As shown in Figure 1, the design process in this environment contains four major parts: powerful system-level specification environment based on Matlab, codesign starting from SDL (including design space exploration, refinement steps and functional architecture generation), multiprocessor SoC generation (including prototyping, interface synthesis, RTOS generation and architecture mapping), and co-simulation environment which provides analysis and simulation for each of the design models.

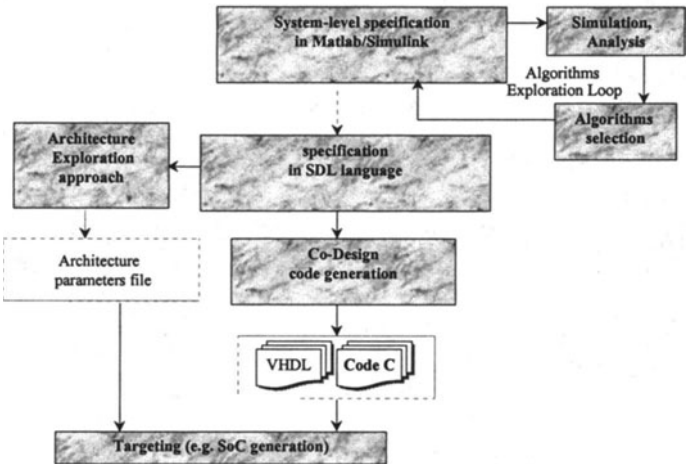

Figure 1: Framework for system validation and prototyping of digital systems

3.1. Algorithm simulation-exploration platform

Many telecommunication systems consist of signal processing and control dominated parts. In order to specify efficiently the full functionality of the system, we propose to use two Matlab and Stateflow. The signal processing part is effectively modeled using a Matlab whereas a Stateflow is better suited for the control logic. Matlab is an intuitive language technical computing environment which provides core mathematics, engineering functions and advanced graphical tools for data analysis, visualization, and algorithm and application development. It is very suited for representation of data flow oriented design. Stateflow provides a key solution for designing the control or protocol logic found in embedded system. Simulink is a simulation environment which provides a block diagram interface that is built on the core Matlab. It also provides an elegant solution to integrate Stateflow blocks for designing the control or protocol logic found in embedded system.

Algorithm simulation-exploration platform is an integrated product suite that combines algorithm design, block diagram simulation, code generation, and analysis in a single, interactive environment. Based on Matlab, Stateflow and Simulink, the

platform shortens development cycles and reduces the risk of design errors by solving many of the engineering problems that he encounter every day, including ambiguous design specifications, reducing the burdensome amounts of manual re-coding, changing the models from one design tool to another and coping with the expensive delays and reiterations caused by design errors. It enhances designer's ability to investigate new research ideas and design custom solutions to complex problems.

The first step a designer might take in creating a high-level executable specification of telecommunication system using simulation-exploration platform is to model the algorithms as a hierarchical mixed data/control flow diagram. The second step is to find the most appropriate algorithm for each part of system and to validate the full system specification. Assuming that the model behaved as expected, a designer would now choose the appropriate architecture and whether to implement the system in hardware or software. These issues will be solved in the following sections.

3.2. Architecture exploration, refinement steps and code generation

In order to use the SDL-based codesign tool and architecture exploration approach, the initial high-level executable specification will be transcribed to SDL.

3.2.1. Architecture exploration. The architecture exploration technique is used here to provide feedback, which guides the search for good architectural solutions. The main issue is to find the most appropriate multiprocessor architecture, including determination of the right partition between hardware and software components and selection of the right components, as well as the best communication protocols. The approach combines both system and RT levels, and uses a hybrid analytic/dynamic model, which gives a good trade-off between speed and accuracy. The architecture exploration approach makes use of the system-level simulator and the hardware/software codesign tool. Figure 2 shows the four steps of this approach: calculation of the basic delays, back-annotation, selection of the architecture and simulation.

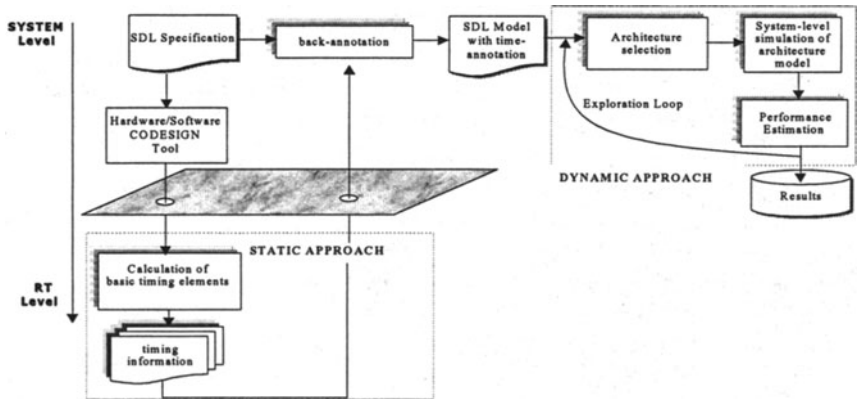

Figure 2. Architecture Exploration approach.

3.2.2.refinements steps and code generation. The process starts from the system-level specification language SDL to produce a functional prototype architecture composed of hardware and software components. The required system functionality is specified in SDL. The second step consists on hardware/software partitioning. The partitioning represents the mapping of functional subsystems onto abstract processors. The third step consists on communication refinements including protocol selection and primitives insertion. The next step examines design alternatives to identify those that meet the system constraints and the architectural choices are made with the assistance of the architecture exploration technique. Then the codesign tool carries out the generation of the functional prototype (including C and/or VHDL codes generation) which is validated through co-simulation.

3.3. Targeting and multiprocessor SoC generation

The targeting will proceed from the top level SoC architecture and the individual components contained on the SOC. The software is partitioned into low-level device drivers interfacing with hardware components, a real-time operating system (i.e. RTOS) and protocols stacks, and application software. The main step here is the design of the communication. The communication interfaces and interconnect components is realized in dedicated hardware. The SoC architecture may contain memory model, processors and peripherals, and application specific cop-processor. In this stage, the architectural model is more detailed and the cycle-true validation is obtained through slow co-simulation. For processors, a BFM will be used in conjunction with an instruction set simulator (ISS). The hardware (RTL-VHDL) will be executed by VHDL simulator (VSS). The cosimulation environment ensures the synchronization of the running simulators for coherently execution of the overall system.

4. Case study

To demonstrate the efficiency of the fast prototyping system for telecommunication applications, we designed a vocoder system. The most prominent vocoder standard is the U.S.Government Linear Predictive Coding vocoder (LPC-10) standard which operates at 2.4 kbps and has been widely used in the military and wireless communication. The following sections describe the basic principles and key features of LPC-vocoder, including functional structures and signal-processing flow in each major module as well as the overall design/validation cycle of LPC-vocoder 2.4 kbps.

The model assumes that speech is the result of exciting the linear time-varying filters (including the LPC filter, and the pitch filter) with a source signal. The excitation source signal is modeled as either a periodic impulse train for voiced speech like vowel sounds, or a random noise for unvoiced speech like consonants. Figure 3 shows the functional block diagram of the vocoder. The vocoder is composed of two major parts: transmitter and receiver where each part contains several algorithms. In the transmitter, the original speech is partitioned into 20-ms frames, each consisting of 160 samples at a sampling rate of 8 kHz. The transmitter

analyzes the original speech, and extracts a set of parameters that represent some kind of source-filter model. These parameters are then transmitted out to the receiver, where a synthesizer reconstructs the speech based on the received parameters.

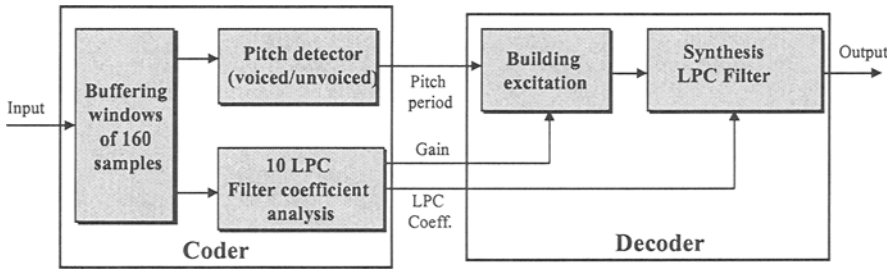

Figure 3. The functional block diagram of the LPC 2.4-kbps vocoder

The simulink specification of LPC vocoder is shown in Figure 4(a). In transmitter, as shown in Figure 4(b), each frame of input speech goes through a high-pass filter and a Hamming window filter, before LPC-analysis filter algorithm is performed. The goal of LPC-analysis filter is to search for a set of optimal filter coefficients, in the sense of the least mean squared residual error. In the LPC filtering, the auto-correlation coefficients of input speech are calculated, then Durbin's recursion algorithm is used to compute 10 optimal LPC coefficients and gain. After LPC analysis, the next step is pitch search which performed by the pitch detector block including pitch period computation and voiced/unvoiced speech detection. Figure 4(c) shows the receiver simulink specification. The excitation block building impulse train according to the pitch period (when the signal was voiced speech) or randomly period (when the signal was unvoiced) as noise. this impulse train get in to the inverse filter that produced according to LPC coefficient and gain. The LPC inverse filter output the reconstructed sample.

4.2. LPC-Vocoder simulation-exploration platform

Figure 5. shows the simulation-exploration platform of LPC-vocoder based on Simulink environment. It consists of exploring Algorithms including algorithm development, analysis and validation. In our case, we have tried 3 pitch extraction algorithms including auto-correlation, center clipping and sift algorithms. The model execution has three phases. The reset phase, parameter values are obtained and checked and the model is initialized. In the main phase of execution, the model reads and writes data. In the final phase, a wrap-up occurs, for freeing resources, writing final results, etc. However, all instances are reset at the beginning of simulation run, and all are "wrapped up" at the end.

In this platform we can either record our own voice via microphone or load samples of prerecorded speech. The next steps are the LPC and the pitch analysis. Both, the set of LPC coefficients and the pitch values are then stored in the parameter memory. These parameters are needed to control the synthesis part of the vocoder which is shown in the lower part of the diagram.

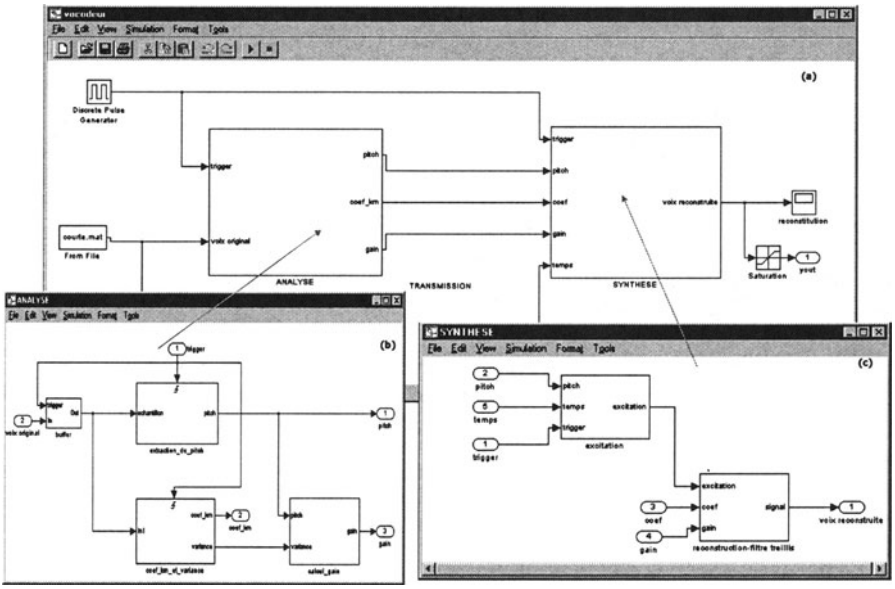

Figure 4. LPC 2.4kbps-Vocoder Simulink specification

The pitch values (pitch contour) and the number of prediction coefficients can be changed and these changes have a significant influence on the reconstructed speech. We can replay the signal and compare it with the reference speech signal. For visual comparison the reference speech signal and the reconstructed speech signal are depicted in both the time and frequency domain. For the reconstructed signal, also the pitch frequency contour is graphically presented and we can directly manipulate this contour. The main advantages of the simulation-exploration platform are its numerous interactive functions.

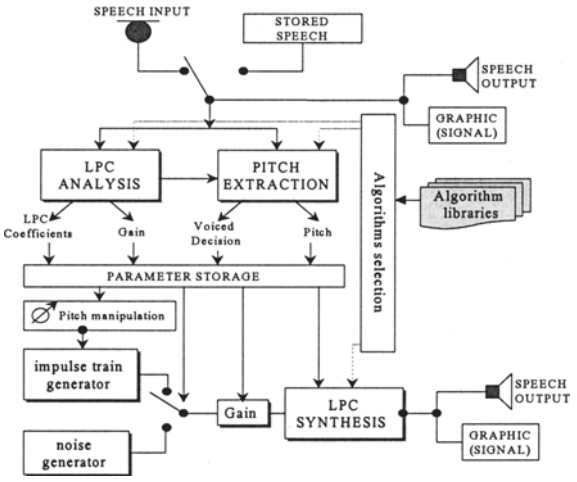

Figure 5. Simulation-exploration platform of LPC-vocoder

4.3. Transcription of the Simulink specification in SDL

During the previous phase, the simulink specification of vocoder with was carried out and validated. We point out that the refinement steps in our framework support only the SDL specification. In order to proceed to codesign and architecture exploration phases, the initial specification is translated into SDL. We have adopted a systematic method to translate the initial specification into SDL, in order to preserve the same structure scheme. Each simulink block is decomposed into SDL block or SDL processes. We have obtained the same result between SDL and Simulink specifications including the blocks hierarchies and sub-blocks. The only difference appears on the SDL processes. This is shown in Figure 6, were the terminal blocks of simulink are translated into processes. The validation of this specification is performed using SDL simulator.

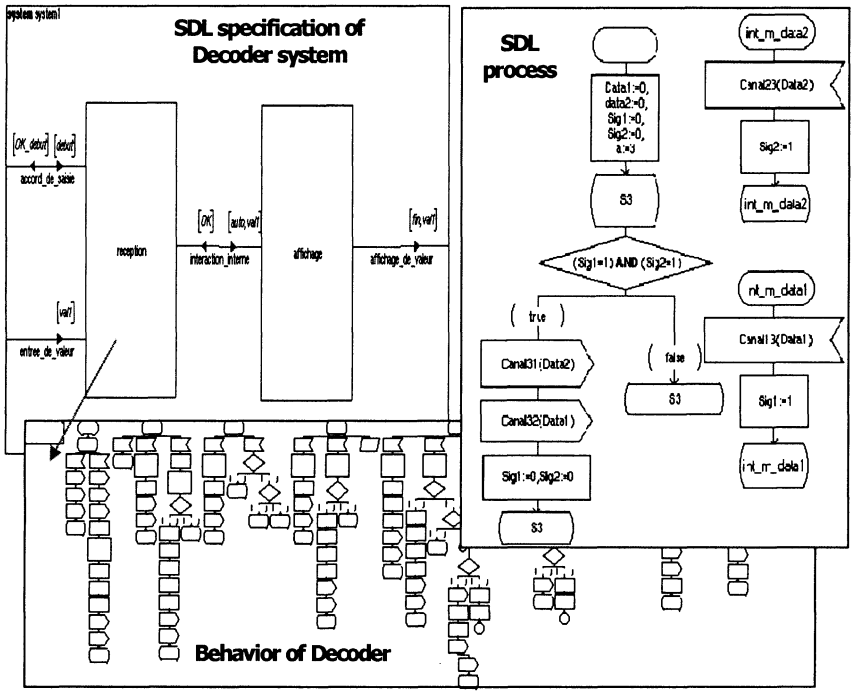

Figure 6. SDL specification of LPC 2.4kbps Decoder

4.4. LPC-Vocoder steps refinement and code generation

The next steps make the refinement in this new system specification. In the first step, we have translated the SDL specification into an intermediate format containing an extended FSMs, in order to apply system refinement transformation. This translation is performed automatically. We have used, then, the interactive partitioning to achieve the different transformations such as the reordering blocks and processes

within the hierarchy and merged into single block or process. We have chosen to implement the all vocoder modules into software. Then we have generated six C code blocks (i.e. processes). We have assigned two processes to one abstract processor and four processes to an other one. After this step, it becomes possible to make a cosimulation of this functional prototype, in order to validate the above refinement steps.

4.5. LPC-Vocoder SoC generation

We chose two kinds of processors: ARM7 micro-controller and TMS30C31 DSP. The partitioning and the assignment of processors for this architecture is illustrated by the Figure 7. The analysis filter and The inverse filter blocks are implemented in software on TMS30C31 DSP. The voice handset and the communications protocol, pitch detector, the voiced/ unvoiced decision, and the excitation generator blocks are implemented in software on ARM7. The communication interfaces and interconnect components is realized in dedicated hardware. The virtual prototype validation is obtained through cycle accurate cosimulation. For processors, a BFM has been used in conjunction with an instruction set simulator (ISS). For hardware, RTL-VHDL has been used and executed by VHDL simulator (VSS). Figure 8 shows the cycle accurate cosimulation of the LPC-Vocoder system.

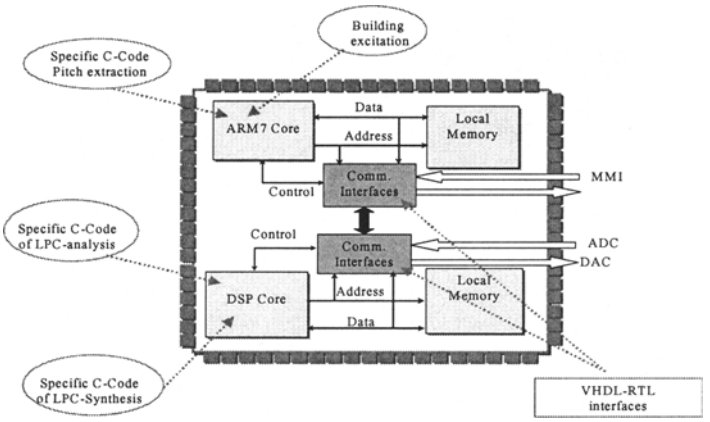

Figure 7. The SoC prototype of the LPC-Vocoder.

5. Result analysis and lessons learned

Table 1 summarizes the simulation and co-simulation time and design facilities. The Simulink description including algorithm exploration and system level simulation of LPC-Vocoder took less then 1 weeks. While SDL specification and validation took less 3 weeks. The difference is due the lake arithmetic operations to perform the signal processing functions in SDL. The architecture exploration and prototype generation and validation took less then 1 week. Considering the simulation speed, in the system-level specification, the simulation time of the Simulink model is 5 time

faster than SDL specification. The difference is mainly due to the overhead procedures dedicated to compute the mathematical functions. The co-simulation of the functional prototype is 24 time faster than cycle accurate cosimulation. This difference demonstrates clearly the benefit of the validation phase of the functional prototype. The simulator of the TMS320C30 runs at speed of 1mega cycles per seconds and the ARM7 runs at a speed of 1.4 mega cycles per seconds.

This experiment shows that Simulink is very suited for the system-level specification and algorithm exploration of DSP applications (i.e. easy description and faster simulation, powerful algorithm exploration, thank to Matlab toolbox). However, the main restriction now, is the lack of design automation tools starting from Simulink. From the SDL point of view this experiment shows clearly the efficiency of the SDL-based design tool including architecture exploration. However, the experiment shows also that is very hard to describe the DSP application in SDL, where the set of predefined arithmetic operation is quite restricted. Of course, SDL remains is very suited for the specification of telecommunication protocols.

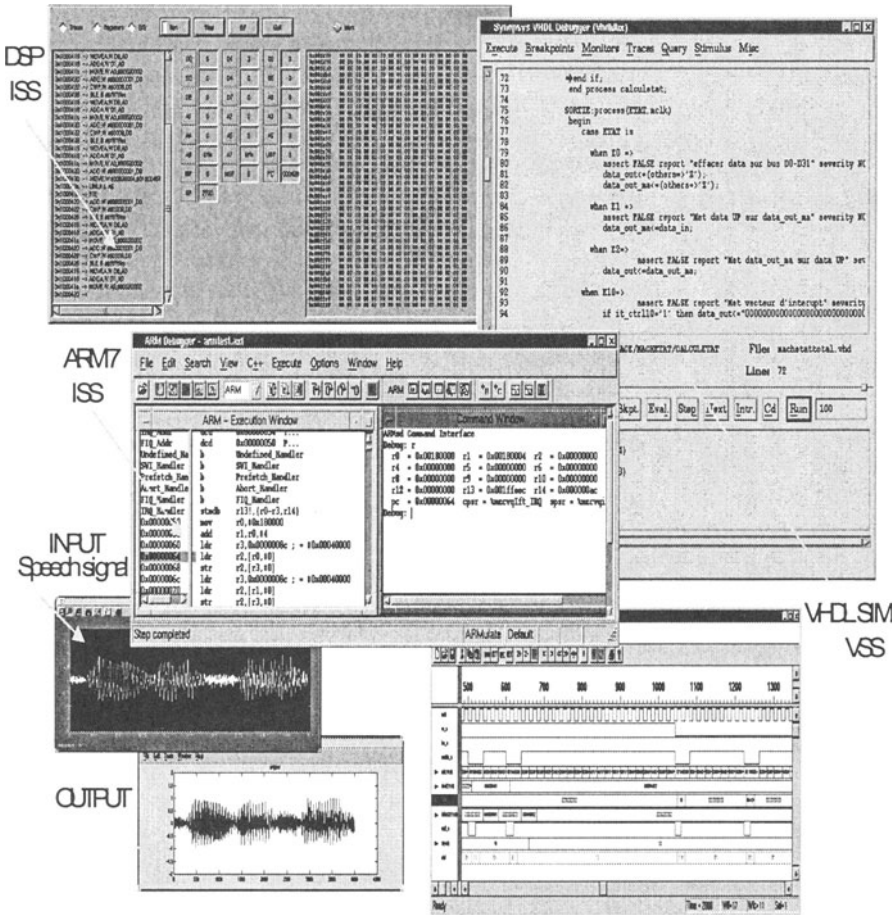

Figure 8. Cycle accurate cosimulation of the LPC-Vocoder system.

Table 1. Simulation and co-simulation time of LPC-Vocoder.

Description Style of Vocoder	Simulation and cosimulation Time of 2 Seconds of speech	Design facilities
Matlab	4 minutes	****
SDL	20 minutes	*
Functional Architecture (multi-module C)	~ 30 minutes	***
Multi-processor (ARM-ISS, DSP-ISS, VHDL)	~ 12 hours	***

6. Conclusion

In this chapter, we have presented a methodology and framework for system design, validation and fast prototyping of advanced telecommunication multiprocessor System–On-Chip. The presented approach takes into account system-level specification, multi-level validation, both algorithm and architecture explorations, refinement, functional architecture generation software and hardware design, and final prototyping generation. Our design framework combined a different technologies such as Matlab, SDL and codesign tool, as well as other related tools, to support all development cycle. The originality of this approach is mainly the combination of an algorithm and architecture explorations in the single design flow. We have also presented the design and validation of the 2.4 kbps-LPC vocoder using our framework. The successful design and result of this case study demonstrated the effectiveness of our approach.

References

[1] COSTE P., et Al., (1999) "Multilanguage specification for the design of heterogeneous systems", *IEEE Hw/Sw Codesign Workshop CODE-99*, Italy.

[2] Eikerling., et Al., (1996)"A Methodology for Rapid Analysis and Optimization of Embedded Systems" IEEE International Symposium on ECBS, D-Friedrichshafen, pp. 252-259, March.

[3] GAJSKI D., VAHID F., NARAYAN S., GONG J., «System-Level Exploration with SpecSyn» *Design Automation Conference*, pp. 812-817, June 1998.

[4] D. Gajski, et Al. SpecC: Specification Language and Methodology, Kluwer Academic Publishers, Boston, MA, ISBN 0-7923-7822-9, March 2000, 336 pages.

[5] R. K. Gupta and G. De Micheli, A Co-Synthesis Approach to Embedded System Design Automation, Design Automation for Embedded Systems, 1(1-2), pp. 69-120, Jan. 1996.

[6] Balarin.F et Al. Hardware-Software Co-Design of Embedded Systems, The POLIS approach, book Kluwer Academic Publishers, 1997.

[7] HENKEL J., ERNST R., « High-Level Estimation Techniques for Usage in Hardware/Software Co-Design» *Asia and South Pacific Automation Conference*, pp. 353-360, Yokohama, Japan, February 1998.

[8] HERRMANN D., HENKEL J., ERNST R., «An Approach to the Adaptation of Estimated Cost Parameters in the COSYMA System» *International Workshop on Hardware-Software Codesign*, pp. 100-107, 1994.
[9] R. Ernst. Hardware/Software Co-Design of Embedded Systems. In Asia Pacific Conference on Computer Hardware Description Languages (APCHDL 97), Hsin-Chu, Taiwan, August 1997.
[10] MADSEN J, et Al, "LYCOS: the Lyngby Co-Synthesis System" *Journal for Design Automation of Embedded Systems, Kluwer.* vol.2, no.2, pp. 195-235.
[11] Hardt, W. Rettberg, A. Kleinjohann. B. (99), The PARADISE Design Environment", in Proceedings of the 1st New Zealand Embedded System Conference 1999. Auckland (New Zealand), July 1999.