Design Methodology of Signal Processing Algorithms in Wireless Systems

P. Belanović, M. Holzer, D. Mičušík, and M. Rupp
{pbelanov,mholzer,dmicusik,mrupp}@nt.tuwien.ac.at
Vienna University of Technology
Institute for Communications and RF Engineering
Christian Doppler Pilot Laboratory for Design Methodology of Signal Processing Algorithms
Gusshausstr. 25/389, 1040 Vienna, Austria

ABSTRACT

Design of modern communication systems is increasingly inefficient using current design methodologies. Significant increases in efficiency, reduction of time to market and improvement in quality can be achieved by adopting a consistent design methodology. Such a design process, based on a single system description resident in a database and integrating all tools used by all design teams is proposed. An implementation of the single system description database and a tool chain based on SystemC is presented. Also shown are the results of processing a real-world wireless communications algorithm through this tool chain.

Keywords: Design Methodology, Signal Processing and Wireless Communications.

1 INTRODUCTION

Modern wireless communication systems require the deployment of highly sophisticated signal processing algorithms and increasingly complex protocols in ever shorter time periods[1]. Implementations of these systems are increasingly heterogeneous, incorporating diverse hardware components such as DSPs, FPGAs and ASICs, as well as software components at various abstraction levels, written in assembler, C, C++, Java, SystemC and similar languages.

Such highly complex systems cannot be efficiently developed using current design methodologies. On the one hand, algorithmic complexity of these systems, which grows according to Shannon's Law, increases faster than the available computational power, which grows according to Moore's Law. On the other hand, in what is termed the ”design gap”, the growth of the available computational power outpaces the growth of design efficiency[2], as shown in Figure 1.

2 DESIGN METHODOLOGY

The design process, leading from concept to realization, passes through three general levels of refinement, namely the *algorithmic*, the *architectural* and the *implementation* levels. Typically, three separate teams can be associated to one of these stages each[3], as shown in Figure 2.

In the design process, the three teams have necessarily distinct areas of expertise, to tackle each of the three stages. Inherently, each of the teams works with a dedicated set of tools on a system description that is optimized for its work.

2.1 Shortcomings of the Current Methodologies

As described above, current design methodologies have several shortcomings. Firstly, descriptions of the system at the three stages of the design process are fundamentally different, making forward and backward communications between teams highly difficult. Consequently, system descriptions are constantly reformatted and rewritten by the corresponding experts to incorporate input from the other teams. This mode of operation is error-prone, slow and inefficient.

Furthermore, the impeded communications between teams can lead to delayed discovery of design faults. As a rule of thumb, a fault that produces a cost of €1 when found at the algorithm level will produce a cost of €6 when discovered at the architectural level and €100 at the implementation level[4].

All the mentioned drawbacks of the current design process are especially severe in the wireless communications field. Increasing complexity of algorithms, such as in UMTS and HyperLAN/2, as well as the extremely tight time to market and cost requirements, together place a great burden on the design process. Complex designs must be produced quickly and correctly the first time.

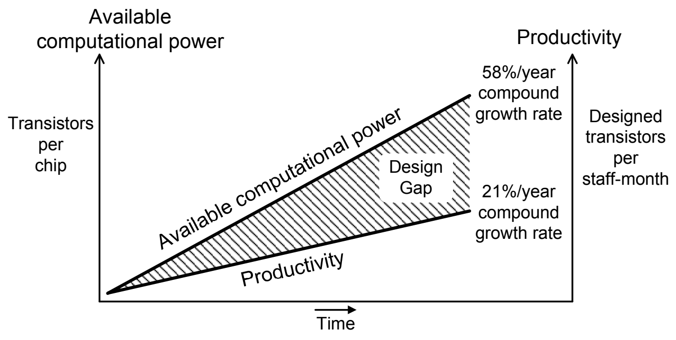

Figure 1: The growth of the design gap

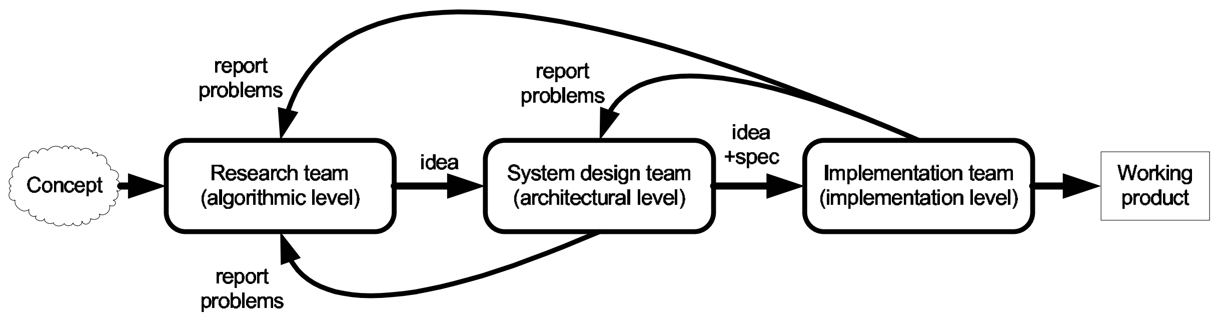
report problems
report problems
report problems
Concept
Research team (algorithmic level)
idea
System design team (architectural level)
idea +spec
Implementation team (implementation level)
Working product

Figure 2: Typical product development process

2.2 A Consistent Design Process

Clearly, significant increase in efficiency, reduction of time to market and improvement in quality can be achieved by providing a consistent design process. Such a process would provide a unified design environment, supporting all the teams equally and allowing them to work on a single system description. Thus, each team would apply its expertise and refine the single system description on its way from concept to realization, but at any point in time each team will have insight into the current description of the system, without translation, thus avoiding any communication obstacles.

In this improved process, each team still requires its usual set of tools. Hence, the unified design environment will have to seamlessly integrate all the existing tools required by all the three teams, as they are provided by many EDA tool manufacturers (for example, SPW from CADENCE, CoCentric System Studio form SYNOPSYS, or N2C from CoWare). The most important aspect of this integration is the binding of each tool to the single system description. Since there exists only one system description, but a great variety of tools, each with unique requirements, means of providing inputs for and incorporating outputs of all the tools must be provided.

Even when all the currently available tools used by the three teams are integrated into a unified design environment, significant steps in the design process would not be covered. These are the steps that are currently carried out manually and only made possible by the expertise and experience present in the three design teams. Completeness of any consistent design process hinges on its ability to allow these design steps to be performed on the single system description, either directly and manually by the corresponding experts, or automatically by newly available dedicated tools. Examples of such significant, yet manual or badly supported tasks are hardware/software partitioning, architecture mapping, bitwidth optimization and others.

3 PROPOSED FRAMEWORK

A flexible, expandable, secure and fast implementation of a single system description is in the form of a database. Such an implementation, operated on by a set of tools, each with dedicated ”in” and ”out” porting software is shown in Figure 3. The three design teams provide inputs, such as desired system behavior and structure, constraints, tool options and others. Also, the designers receive outputs, such as status of the system description, results of simulations, estimates of hardware costs, timing and similar. Some of the tools are those currently used by the design teams, while others are specially written to perform missing tasks, either automatically or manually by designers, as described in Section 2.2.

Such special tools are initially written as database modification tools that simply allow the designer to enter manually derived values. The database is thus enriched and the system description is refined on its way to implementation. For example, a tool may allow the designer to specify implementation in hardware or software for each system module in the database. Hence, the designer performs manual hardware/software partitioning.

However, the proposed consistent design process not only allows such manual modifications to be performed more quickly and with less chance for error than they are currently, but it also provides for an easier environment to automate these procedures. Then, for example, the designer may not need to

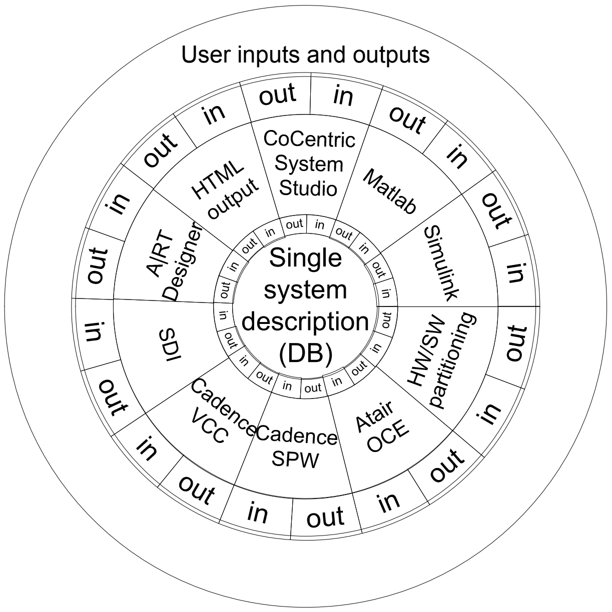

Figure 3: Proposed consistent design process

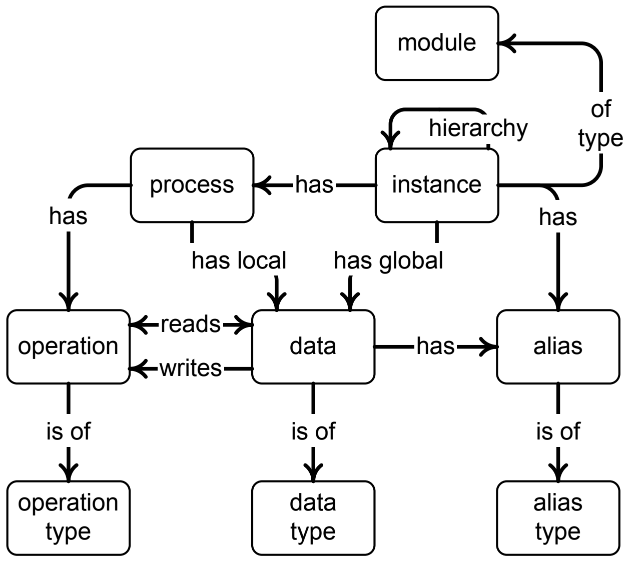

Figure 4: Structure of the design database as an entity-relation diagram

manually enter bitwidth information for each signal, but rather just provide necessary information to a range propagation algorithm which enriches the database with bitwidth information automatically.

4 IMPLEMENTATION

An implementation of the design methodology presented in Section 2.2, supporting a partial design flow based on SystemC, has been achieved. This implementation relies on a single system description, presented in Section 4.1. Importing of the SystemC code describing the system into the database is handled by the System Description Investigator (SDI), presented in Section 4.2. A dynamic viewer of the single system description has also been implemented, as described in Section 4.3.

4.1 Single System Description

The single system description is the central repository of the consistent design environment described in Section 2.2. All tools integrated into the environment are bound to the single system description, as shown in Figure 3. The implementation of the single system description presented here is achieved through a MySQL database, running on a Sun Blade 2000 server.

Structure of the implemented database is shown in Figure 4. This structure has been designed to generally fit system descriptions, with support for such concepts as modules or entities, their hierarchy and interconnections. In addition to such concurrent concepts, sequential parts of system descriptions, such as processes and operation sequences are also supported. The nomenclature of all concepts in the database structure implementing the single system description follows that of the SystemC language.

All entities that make up the system are *instances* of *modules*. These instances form one or more layers of hierarchy. Each of the instances can contain one or more *processes*. All processes in the system run concurrently. Processes are internally sequential, formed by sequences of *operations*. Communications between instances, processes and operations is achieved through *data*. Data connecting several instances has several *aliases*; one within the context of each of the connected instances. An alias has an *alias type*, such as input, output, in-out port or internal signal. Data has a *data type*, such as a signal, variable or constant. Operation also has an *operation type*, such as addition (+), multiply-accumulation (MAC) or left bitwise shift (<<).

4.2 System Description Investigator (SDI)

Creation of the database structure described in Section 4.1, as well as the importing of a SystemC system description into the database is achieved through the SDI module. This software module is composed of two parts: *parser* and *scanner*.

The parser is responsible for parsing the SystemC code representing the system and extracting all relevant information, such as module instantiations, signal interconnections and port declarations. Information extracted by the parser is stored in an Intermediate Format (IF) file. The parser is based on lexical and syntactical analysis using Flex and Bison.

The scanner is a set of Perl scripts that manage the database structure and import the information stored in the IF file. After creating the database structure, the scanner scripts relate all the pieces of information in the IF file, extracting meaningful concepts, and store them in the database.

4.3 HTML Visualizer

To provide the designers working on the development of the system whose description resides in the database with a visual representation of the system, an HTML Visualizer has been developed. This software module, implemented as a Perl CGI script running on an Apache web server, provides an easily extendable viewer of the database contents. Since it is a CGI script, the HTML Visualizer provides a dynamic view of the system description, immediately reflecting any changes and updates. Both textual and graphical information representing the database content is shown by the HTML Visualizer. Content of the database tables are displayed tabularly and the hierarchy of instances in the system is represented graphically.

5 WIRELESS COMMUNICATIONS EXAMPLE

As an example of a wireless communications algorithm, a SystemC 1.0 (register-transfer level) [5] description of a cell searcher algorithm has been successfully processed by the tool chain. This algorithm implements the slot synchronization of user equipment in a UMTS system. As seen in Figure 5, the cell searcher model consists of a chain of blocks.

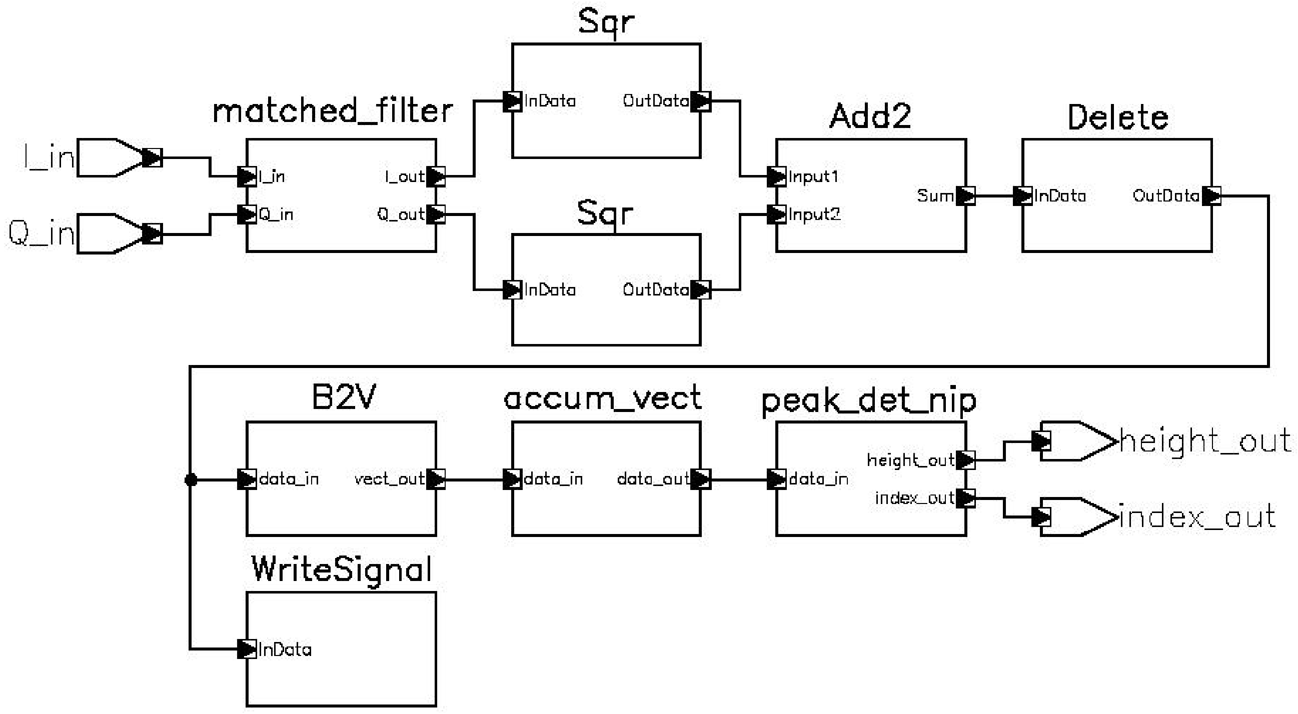

Figure 5: Cell searcher model

The first is a matched filter, followed by two parallel instances of the arithmetic squaring function and an addition block. Following these is the block that deletes the first element in the data chain (necessary in the simulation of the system). Next is the Bit-to-Vector converter that parallelizes the data stream. Following this are the accumulator, which sums all the incoming vectors, and the peak detector, which looks for the position and the value of the peak in the data stream.

Benchmarked for 100 runs of the tool chain, the parsing of the cell searcher system description took 0.169μs, whereas the database management took 38.66μs.

6 CONCLUSIONS

Provision of a single system description and a unified design environment to all teams in the design process has the potential to improve efficiency, reduce time to market, cut costs and improve quality. Integration of the current design process to operate on a single system description requires porting all existing tools used by the designers to this system description. To achieve the remaining tasks, dedicated manual accesses to the system description need to be provided. Finally, these manual accesses can be automated, providing a complete and consistent design process.

ACKNOWLEDGEMENTS

This work has been funded by the Christian Doppler Pilot Laboratory for Design Methodology of Signal Processing Algorithms.

References

[1] V. N. Patel, C. K. Wiese, F. M. Hiemstra, and S. Himes, “Rapid Development and Commercialization of Products-A Business Imperative in the Global Telecommunication Landscape,” *Bell Labs Technical Journal*, pp. 157–171, October-November 2000.

[2] International SEMATECH. www.sematech.org.

[3] M. Rupp, A. Burg, and E. Beck, “Rapid Prototyping for Wireless Designs: the Five-Ones Approach,” *Signal Processing Europe 2003*, June 2003. (accepted for publication).

[4] R. S. Janka, *Specification and Design Methodology for Real-Time Embedded Systems*. Kluwer Academic Publishers, 2002. ISBN 0-7923-7626-9.

[5] Open SystemC Initiative. www.systemc.org.

[6] T. Grötker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Kluwer Academic Publishers, 2002. ISBN 1-4020-7072-1.