C- BASED RAPID PROTOTYPING FOR DIGITAL SIGNAL PROCESSING

Emmanuel CASSEAU, Bertrand LE GAL, Pierre BOMEL, Christophe JEGO*,
Sylvain HUET and Eric MARTIN

LESTER lab, CNRS FRE2734, UBS University, France, http://lester.univ-ubs.fr:8080
* CNRS TAMCIC, GET/ENST Bretagne, France, http://www.enst-bretagne.fr

ABSTRACT

The increasingly demanding requirements of digital signal processing applications like multimedia, new generations of wireless systems, etc. led to the definition of more and more complex algorithms and systems that are to be efficiently implemented with the time to market constraint. Today, the electronic system design community is mainly concerned with defining efficient System-on-a-Chip (*SoC*) design methodologies in order to benefit from the high integration capabilities of current *ASIC* and *FPGA* technologies on the one hand, and manage the increasing algorithmic complexity of applications on the other hand. Rapid prototyping is considered as a key to speed up the system design. In this context, we have introduced a novel methodology that efficiently addresses both the algorithmic complexity and the high flexibility required by the various application profiles. Our methodology benefits from the emerging High-Level Synthesis (HLS) tools in a platform-based approach dedicated to the rapid prototyping of real-time systems. We show the effectiveness of this approach with the design of a DVB-DSNG compliant receiver.

1. INTRODUCTION

Signal and image processing systems are facing heavy constraints due to the more and more demanding requirements of applications including computing complexity and various application profiles. Flexibility is usually addressed through software solutions where a variety of programs can be downloaded into a fixed processor-based platform. However, general-purpose processors or digital signal processors have limited computation speed due to the limited number of processing units available for parallel computation. When higher computation speed is required, mixed hardware/software solutions are usually preferred. However, hardware design is not flexible, time consuming, and consequently incompatible with both time to market constraint and efficient design space exploration

As semi-conductor very deep sub-micron technologies ever get deeper, platforms and prototypes have become important concepts in the design and validation of electronic systems to provide correct-the-first-time SoCs. Furthermore observing that many of the functions are well known and have already been implemented, the system design flow can be dramatically accelerated by re-using blocks instead of re-designing them from scratch. Current design trends give priority to hardware and software re-use through *Virtual Component* (*VC*) or *Intellectual Property* (*IP*) exchange [1,2]. In this context, platforms aim at providing an IP-reuse framework for SoC design, thus reduce the IP development and integration phases.

This paper presents our contribution in the field of digital signal processor high-level synthesis for a platform-based approach dedicated to rapid prototyping. High-level synthesis of hardware intellectual properties frees designers from designing custom processor cores to reach the application efficiency required or implementing time consuming specific interfaces. Hence, it shortens prototype refinements and then allows a wider system space validation by rapid prototyping instead of pure (co)simulation or emulation.

The paper is organized as follow: in section 2 we describe our contribution in term of rapid prototyping platform design and synthesis target using high-level synthesis. In section 3, we describe the rapid prototyping of a radio communication system involving the DVB-DSNG (Digital Video Broadcasting - Digital Satellite News Gathering) standard. Conclusion is presented in section 4.

2. RAPID PROTOTYPING SYSTEM DESIGN FLOW

2.1. System design flow

In the prototyping platform context the design flow consists in mapping the functional architecture (interconnected algorithmic functions, e.g. C/C++, Matlab or Ptolemy specifications for example) of the application to be implemented onto the targeted platform architecture. The functional architecture model is thus mapped onto a heterogeneous HW/SW platform, as represented on figure 1. The software code generation is usually performed using a compiler that compile a C description into the assembly code that runs on the targeted processor. For the hardware parts of a SoC, two design approaches are currently used:

- Hand written design at the RT (Register Transfer) level allows optimal performance but is associated to important development and verification time,
- IP core including the design of a wrapper (communication interface) in order to satisfy from the one hand the system constraints and the IP requirements from the other hand [3].

Our approach consists in reducing the hardware IP core development time by benefiting from the emerging High-Level Synthesis (HLS) tools. Our methodology aims at facilitating design, validation and synthesis of IP cores at

the behavioral level, and exploits functional as well as architectural flexibility by allowing straightforward instantiation of various RTL architectures – fulfilling various sets of functional parameters and performance constraints – starting from a single high-level description of the behavior.
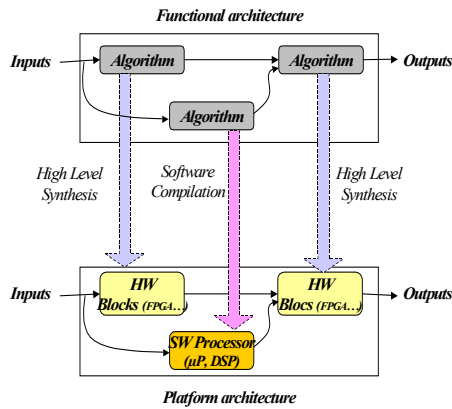

Figure 1 Architecture mapping including high-level synthesis

2.2. Introduction to high-level synthesis

High-level synthesis [4,5] is analogous to software compilation transposed to the hardware domain. The source specification is written in a high-level language (Matlab, C, SystemC, etc.) that models the algorithmic behavior of a complex hardware component. An automatic refinement process allows the mapping of the described behavior onto a specific technology target depending of targeted constraints. A design flow including high-level synthesis thus allows fast algorithm implementations. Thanks to formal proven automation algorithms, HLS tools generate an RTL architecture which respects the designer and the system constraints and which is reliable (error less) compared to a hand coded design. It claims especially to speed up design time versus register transfer level hand coding.

HLS is a constraint-based synthesis flow: hardware resources are selected from technology-specific libraries of components designed and characterized for a specified target (depending on the targeted platform). HLS can also be constrained to limit the hardware complexity (i.e. the number of allocated resources) and reach a given computation speed. The high-level synthesis refinement process follows a top down approach. Four main tasks are performed: (1) source specification analyze (identify computations); (2) hardware resources selection and allocation for each kind of operation; (3) operation scheduling;
(4) optimized architecture generation, including a datapath and a control finite-state machine.

Thanks to its high abstraction level, a behavioral description for HLS can be made customizable through functional parameters. Each set of supported parameter values and synthesis constraints thus allows to instantiate a different dedicated architecture that will fulfill specific functional requirements and achieve specific performance. As a result, HLS tools can be seen as a relevant approach for

implementing and benchmarking current DSP algorithms on different platforms and hardware resources in a rapid prototyping design process.

A general view of the hardware part design flow including high-level synthesis and behavioral virtual components is shown on figure 1.
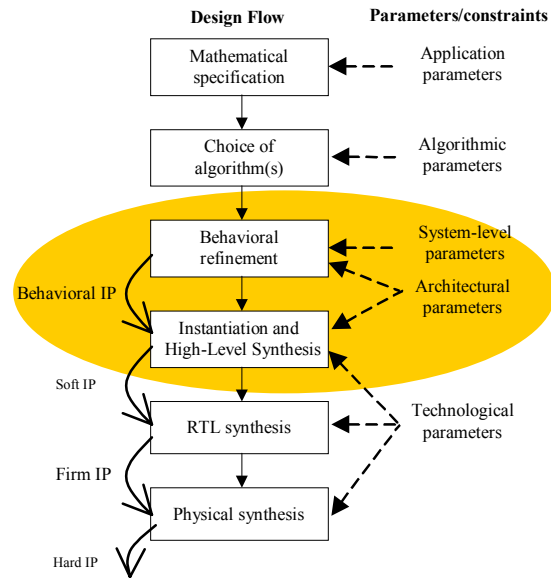

Figure 2 Hardware part design flow based on virtual component reuse

2.3. HLS design context

Many commercial and academic high-level synthesis tools can be used: Catapult-C (Mentor Graphics), AccelFPGA (AccelChip), SystemC Compiler (Synopsys) for commercials and GAUT, SPARK, Cathedral, etc. for academics. For our experiments the tool we use is GAUT[1]. GAUT is an HLS tool dedicated to Signal and Image processing applications under real time execution constraints. This tool performs synthesis under latency constraint, memory mapping and data communication consumption/production dates. It thus allows the designer to accurately stipulate the system interaction and constraints with the algorithm to be synthesized. The generated architecture is composed of 3 units: the processing unit (Data-Path and Control-Unit), the memorization unit and the communication unit which sends and receives data from/to the rest of the system. Since rapid prototyping system design is targeted, the architecture is currently generated using formal component libraries characterized for FPGA families (Xilinx and Altera) providing circuit constraint correctness. This architecture is generated in VHDL-RTL (direct input for commercial logical synthesis tools like ISE/Foundation from Xilinx, Quartus from Altera, etc.).

1 GAUT tool is downloadable after a free registration on LESTER web site http://lester.univ-ubs.fr:8080

3. DESIGN EXPERIMENT

The proposed methodology has been used in the ALIPTA (Algorithmic Level IP for Telecom Applications) project. This project aims at developing a receiver compliant with the DVB-DSNG standard (Digital Video Broadcasting - Digital Satellite News Gathering [6]). Maximum cohesion with DVB-S1 is maintained, such as concatenated error protection to improve digital communication quality. In particular, concatenated coding employing an inner convolutional code (decoded by a Viterbi decoder) combined with a Reed-Solomon outer code constitutes an attractive scheme that is commonly encountered in many applications. Transmissions are allowed from 1.5 Mbps to 72 Mbps.

Starting from a functional architecture written in C, the first step of the prototyping approach consists in the mapping of the DVB-DSNG decoding part. Computation metrics at the functional level aims us to make implementation choices for each function. Our mapping scheme is shown on figure 3 where the Viterbi and Reed Solomon decoders are implemented onto hardware blocks (computational intensive functions) and the synchronization part onto a software processor (data control dominated functions).
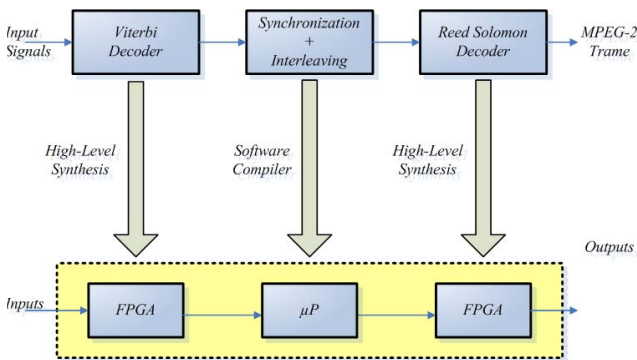

Figure 3 DVB-DSNG decoder mapping

The Sundance platform [7] we used in the ALIPTA project as a rapid prototyping support is composed of the last generation of C6x DSPs from TI and Virtex-E FPGAs from Xilinx. Communications between different functional blocks are implemented with high throughput SDB links (200 Mbytes/s) [7].

The IP core integration can be time-consuming and time-effective i.e. usually requires the design of a specific communication interface due to incompatible protocols and/or asynchronous networks. An automatic generation of the communication interfaces frees the designer from the interface design. We thus have integrated in the HLS process the synthesis of the wrappers to seamlessly interface the synthesized IP with a pseudo-asynchronous communication network and still obtain synthesizable RTL VHDL codes whatever the communication scenario is. At the hardware level the communication between computing nodes is handled by 4-phases handshaking protocols and decoupling FIFOs. The handshaking protocols synchronize computing with communication and the FIFOs enable to store data in order to overcome potential data flow irregularities. Handshaking protocols are used either to communicate seamlessly between hardware nodes or between hardware and software nodes. Handshaking protocols are automatically refined by the GAUT tool to fit with the selected inter-node platform communication interfaces (bus width, signal names, etc …).

The software code of the functions mapped onto software processors is initially written in C. Code Composer from TI is then used to generate the assembly code running on the C62. Platform specific code has to be written to ensure the inter processing elements communication. The communication drivers of the targeted platform are called inside the interface functions through an application programming interface (API) mechanism. Thereby we have developed C++ concurrent processes like I/O drivers.

3.1. HLS-based HW block design

Design synthesis results for the Viterbi and Reed Solomon decoders are presented in the next sections. Results are based on a Virtex-E FPGA technology with a 10 ns clock period, which is the maximum latency of the sequential operators in the technological library.

3.1.1 Viterbi decoding

The Viterbi algorithm is applicable to a variety of decoding and detection problems which can be modeled by a finite-state discrete-time Markov process, such as convolutional and trellis decoding in digital communication [8]. Based on the received symbols, the Viterbi algorithm estimates the most likely state sequence according to an optimization criterion, such as the a posteriori maximum likelihood criterion, through a trellis which generally represents the behavior of the encoder.

The generic SystemC specification of the Viterbi algorithm coming from the C-functional architecture allowed us to synthesize dedicated architectures using different sets of functional/application parameters: state number, survivor path length and throughput. A piece of synthesis results that have been obtained for the processing unit is given in figure 4. With the DVB-DSNG standard, the Viterbi decoder is a 64-states decoder. Different throughput constraints have been tested for this particular case (figure 5) (DVB-DSNG throughput is from 1.5 Mbps to 72 Mbps).

State Number	8	16	32	64	128
Throughput (Mbps)	44	39	35	26	22
Synthesis Time (s)	1	1	3	9	27
Number of logic elements	223	434	1130	2712	7051

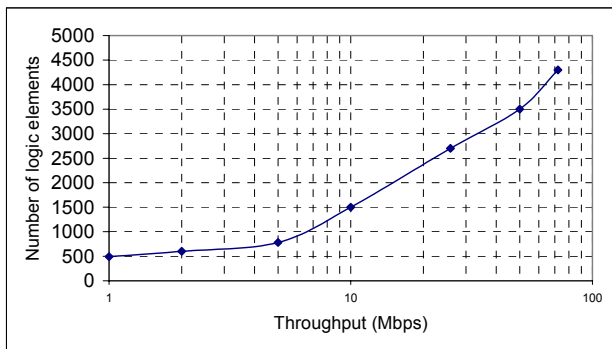**Figure 4 Viterbi decoder synthesis results**

Figure 5 DVB-DSNG Viterbi decoder complexity

3.1.2. Reed Solomon decoding

Reed-Solomon codes are block error correction codes with burst error-correcting capabilities that have found widespread use in storage devices and digital communication systems [6]. The channel coding scheme in the DVB-DSNG uses a (204,188) Reed-Solomon code. This RS code is a punctured version of the RS(255,239) working on bytes. It is able to correct up to 8 erroneous bytes per received packet of 204 bytes.

Thanks to the high level of the input specification of the RS algorithm, several DVB-DSNG RS(204,188) decoder architectures have been generated. Figure 6 gives the complexity (number of logic elements) for different
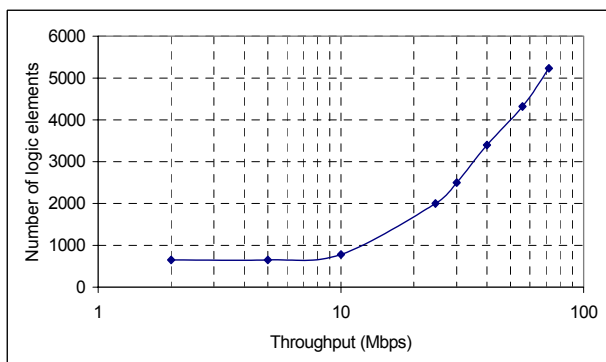

throughput constraints.

Figure 6 DVB-DSNG RS decoder complexity

3.2. DVB-DSNG receiver

In order to validate our methodology, we have implemented onto the Sundance platform the decoding part of a DVB-DSNG compliant receiver with a 26 Mbps throughput constraint. As said previously the generic behavioral SystemC descriptions of the Viterbi and Reed Solomon IP cores have been synthesized with the DVB-DSNG parameter values and this throughput constraint using a HLS tool. Synchronization and interleaving parts have been implemented in software on the C62 DSP. For the interleaving part, a row writing / column reading process has been used with a 204x204 matrix. Data communication is supported by 4 SDB links using instantiated SW

communication drivers and the HLS-based automated HW interface generation. The throughput is bounded by the synchronization computation speed, e.g. the SW part of our design. If higher throughput is required, the DSP has to be changed, for example with a C64 instead of a C62. Thanks to the behavioral level of the virtual components we use, only the real time constraint has to be changed for the high level synthesis of the Viterbi and the RS decoders.

4. CONCLUSION

Traditional methods for rapid prototyping suffer from heavy limitations that prevent them from efficiently addressing both the algorithmic complexity and the high flexibility required by the various application profiles. Our design methodology shows that C-based design associated with high-level synthesis helps the designer in a rapid prototyping objective. It allows fast implementations fulfilling various sets of system constraints (throughput, …) and application constraints (number of coding states, …). It releases the designer from the main implementation constraints and allows fast prototyping for different functional parameters and architecture exploration. This approach has been successfully applied for the design of the decoding part of a 26 Mbps DVB-DSNG receiver we have implemented on a Sundance platform.

5. REFERENCES

[1] M. Keating, P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Design*, 3rd edition, Kluwer Academic Publishers, 2003.

[2] VSIA - Virtual Socket Alliance Interface, http://vsi.org.

[3] J.M. Daveau, G. Fernandes M., T. Ben-Ismail, A.A. Jerraya, *Protocol Selection and Interface generation*, Proceedings of IEEE International Workshop on Hardware/Software Co-Design (CODES), 2001

[4] D. D. Gajski, N. D. Dutt, Allen C-H. Wu, Steve Y-L. Lin, *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers, Boston, MA, 1992.

[5] J. P.Elliott, *Understanding Behavioral Synthesis. A Practical Guide to High-Level Design*, Kluwer Academic Publishers, 2000.

[6] Standard ETSI EN 301 210, *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for Digital Satellite News Gathering* (DSNG) , March 1999.

[7] Sundance Multiprocessor Technology, http://www.sundance.com

[8] A. J. Viterbi, *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Trans. Inform. Theory, vol. IT-13, pp. 260--269, Apr. 1967.