High Level Synthesis Assisted Rapid Prototyping for Digital Signal Processing

Bertrand Le Gal, Emmanuel Casseau, Pierre Bomel, Chirstophe Jégo,
Nathalie Le Héno, Eric Martin

High-Level Synthesis Assisted Rapid Prototyping for Digital Signal Processing

B. Le Gal[1], E. Casseau[1], P. Bomel[1], C. Jego[2], N. Le Heno[3], E. Martin[1]

1. LESTER Lab.– CNRS FRE 2734
UBS University FRANCE
{First-Name.Surname}@univ-ubs.fr
http://lester.univ-ubs.fr:8080

2. ENST Bretagne, FRANCE
christophe.jego@enst-bretagne.fr
http://www.enst-bretagne.fr

3. Turbo Concept SAS, FRANCE
nathalie.leheno@turboconcept.com
http://www.turboconcept.com

Abstract

The increasing needs of higher data rates associated with mobility constraints motivate the development of Digital Satellite News Gathering (DSNG) and Digital Video Broadcasting applications by Satellite (DVB_S). Error control codes like Reed-Solomon and Viterbi codes are widely used in these communication systems against channel noise. Traditional methods for rapid prototyping of hardware cores for this kind of applications are based on RTL specifications However, they suffer from heavy limitations that prevent them from efficiently addressing both the algorithmic complexity and the high flexibility required by the various application profiles in fast implementation and prototyping issues. For this reasons, we propose to reduce hardware IP core development time by benefiting from the emerging High-Level Synthesis (HLS) tools in a platform-based approach dedicated to rapid prototyping. This technique has been successfully applied to the design of Reed-Solomon (RS) and Viterbi decoder IP cores for the DVB-DSNG standard and can be easily extended to many DSP dataflow applications.

1. Introduction

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. As generic terms, platforms and prototypes have meant different things to different people, but are both related to three clear and serious difficulties the electronic industry has to face with: 1) industry disaggregation/horizontalization (market identification, system specification, components development, system assembly, silicon manufacturing, test and packaging are performed by distinct organizations), 2) time-to-market pressure and 3) a dramatic increase in *non-recurrent-engineering* (NRE) costs. These issues are pushing for consolidation of methodologies able to *provide correct-the-first-time* IC designs for high-volume and low-cost *systems on a chip* (SoC). These methodologies are based on systematic component reuse [1] at all abstraction levels and (potentially high-level) synthesis of specific interfaces. They exploit the concept of *orthogonalization of concerns*: a clear separation between architecture design and communication design is an important concept to handle SoC complexity and get productivity gains.

In this context, platforms aim at providing an IP-reuse framework for SoC design, thus reduce the IP development and integration phases. Potential system heterogeneity due to IP reuse is handled by either communication interface synthesis or definition of interface standards the reusable IP must comply with (i.e. point to point interfaces, generic interfaces: OCP, VCI/VSIA, or proprietary buses: AMBA, CoreConnect, PI-bus, etc.).

This paper presents our contribution in the field of *digital signal processor high-level synthesis* (DSP-HLS) for a platform-based approach dedicated to rapid prototyping. We'll demonstrate that the platform-based design *meet-in-the-middle* paradigm fits not only to silicon design, but also perfectly to a rapid prototyping methodology assisted by high-level synthesis of hardware intellectual properties (IPs). It frees designers from implementing time consuming specific interfaces or designing custom processor cores to reach the application efficiency required. 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. We provide in section 3 experimental results of the rapid prototyping of a radio communication system design involving a DVB-DSNG (Digital Video Broadcasting - Digital Satellite News Gathering) system. We finally conclude and give insights in the future research our work suggested us.

2. System Design Flow with High-Level Synthesis

2.1. System Design Flow

In the prototyping platform context the design flow consists in mapping the functional architecture (interconnected algorithmic functions) of the application to be implemented onto the targeted platform architecture. The functional architecture model is thus mapped onto a heterogeneous platform, as represented by the "function mapping" on Figure 1. Actually two design methodologies are currently used for the hardware parts of a SOC:

- Hand written design at the RT level (Register Transfer) 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.

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 such as gate count, speed, etc. – starting from a single high-level description of the behavior.

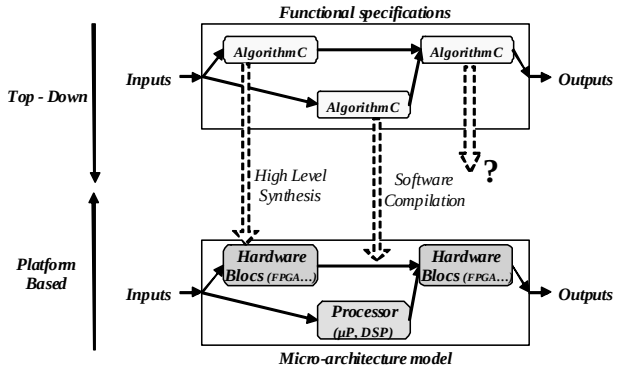

Figure 1 Architecture Mapping including High Level Synthesis

2.2. Introduction to High Level Synthesis

High Level Synthesis [1][2] is analogous to software compilation transposed to the hardware domain. The source specification is written in a high-level language (Matlab, SystemC, C, VHDL, 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 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. HLS can also be constrained to limit the hardware complexity (i.e. the number of allocated resources) and reach a given computation speed. Some HLS tools also support other system synthesis constraints: placement of data in dedicated memorization units [3], data interfacing (arrival/production dates of input/output data) [4]…

The high-level synthesis refinement process follows a top down approach as shown in figure 2. The first synthesis task is the transformation of the algorithmic description to an internal representation model, which captures the algorithmic semantics. From this internal model, four main tasks [2] are performed:

(1) Internal representation analyzing, computation identifying;

(2) Hardware resource selection and allocation: for each kind of operation an operator type and its quantity has to be defined. Operators are selected from the component library defined by the user (components are characterized in terms of gate count, delay, etc.).

(3) Operation scheduling consists in ordering operations on an allocated operator, optimizing hardware reuse, minimizing power consummation and interconnect costs.

(4) Optimized architecture generation, including a datapath, a control finite-state machine and other units if needed (memory units, communication units etc.).

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 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 algorithms on different platforms and hardware resources in a rapid prototyping design process.

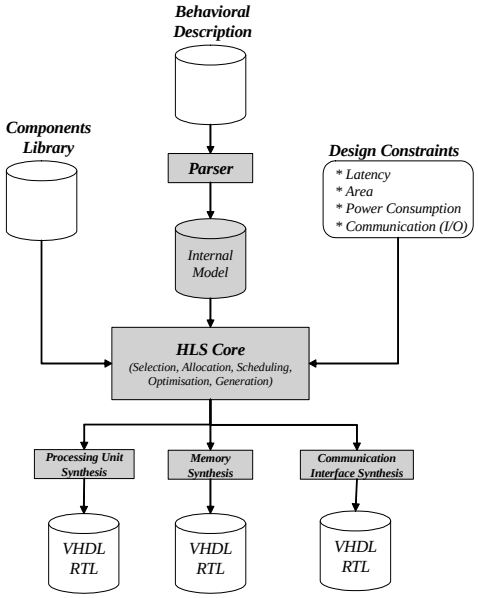

Figure 2 High Level Synthesis Refinement Process

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 pipeline architecture synthesis tool dedicated to Signal and Image processing applications under real time execution constraints. This architectural synthesis tool performs synthesis under

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

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 as shown in figure 3: the processing unit (Data-Path and Control-Unit), the memorization units and the communication unit which sends and receives data from/to the rest of the system. This architecture is generated in VHDL-RTL (direct input for commercial logical synthesis tools like ISE/Foundation from Xilinx, Quartus from Altera, etc.).

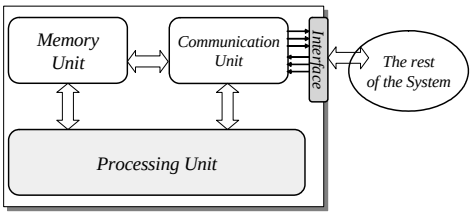

Figure 3 Generated Architecture

As we target rapid prototyping system designs the architecture is currently generated using formal component libraries characterized for FPGA families (more than 190 libraries for Xilinx and Altera FPGAs) providing circuit constraint correctness.

GAUT also provides a VHDL test bench automatic generation which allow the designer to reuse its algorithmic level input stimulus avoiding hand coded errors and harmful time lost in time to market objective.

3. Experimental results

The proposed methodology is currently used in the ALIPTA (Algorithmic Level IP for Telecom Applications) project. This project aims at developing a complete receiver compliant with the DVB-DSNG standard. Maximum cohesion with DVB-S1 is maintained, such as concatenated error protection to improve digital communications quality. In particular, concatenated coding employing an inner convolutional code combined (Viterbi Decoder) with a Reed-Solomon outer code constitutes an attractive scheme that is commonly encountered in many applications (fig. 4).

As said before the behavioral descriptions of these IP cores have been written and synthesized using a HLS tool. From a single behavioral description, a variety of architectures have been generated, spanning a wide range of performance, including the constraints of the DVB-DSNG standard we target.

Before experimenting our approach with the Digital Video Broadcasting and Digital Satellite News Gattering application [5] we first analyze the decoding part. Computation metrics aims us to make implementation choices for each function of the DVB-DSNG decoding part. Our mapping scheme is shown on figure 4 where the Viterbi and Reed Solomon decoders are implemented onto hardware blocs (computational intensive functions) and the synchronization part onto a general processor (software part).

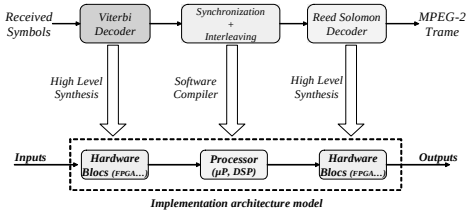

Figure 4 DVB-DSNG Application Mapping

The Sundance platform [6] we used in the ALIPTA project as a rapid prototyping support is composed of the last generation of C6x DSPs and Virtex FPGAs. Communications between different functional blocs are implemented with high throughput SDB links [6]. Automatic generation of communication interface for software and hardware IP frees designer form interface design.

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 (SDB) inter-node platform communication interfaces (bus width, signal names, etc …).

To end the software code generation, 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 introduced in the macro-architecture model through an API mechanism. Thereby we have developed C++ concurrent sequential process like I/O drivers: we provide a specific class for each type of link available on the platform.

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. 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 communications [7]. 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 behaviour of the encoder.

The generic description of the Viterbi algorithm allowed us to synthesize architectures using different values for the following functional parameters: state number and throughput. For each generated architecture, the complexity (amount of logic elements) of the processing unit is given in

figure 5. Note that with the DVB-DSNG standard, the Viterbi decoder includes 64 states.

State Number	8	16	32	64	128
Throughput (Mbps)	44	39	35	26	22
Number of Operations	50	94	182	358	582
Synthesis Time (s)	1	1	3	9	27
Number of logic elements	223	434	1130	2712	7051

Figure 5 Synthesis results for different Viterbi decoders

In the particular case of the DVB-DSNG Viterbi decoder (64 states) different throughput constraints (from 1 Mbps to 50 Mbps) have been tested.

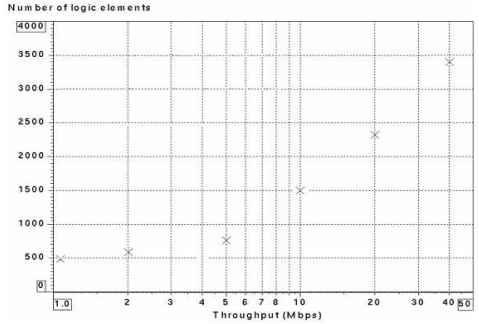

Figure 6 Logic size for different throughputs

3.2. 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 [5]. The channel coding scheme in the DVB-DSNG use 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 [5].

Thanks to the high level of the input specification of the RS algorithm, several RS decoder architectures have been generated using different bit rate values according to different communication standards. In the case of the DVB-DSNG standard transmissions are allowed from 1.5 Mbps to 72 Mbps. Figure 7 gives the complexity of the processing unit of the RS decoder. The complexity is about 650 logic elements until 10 Mbps. It increases until 3500 logic elements for 42 Mbps.

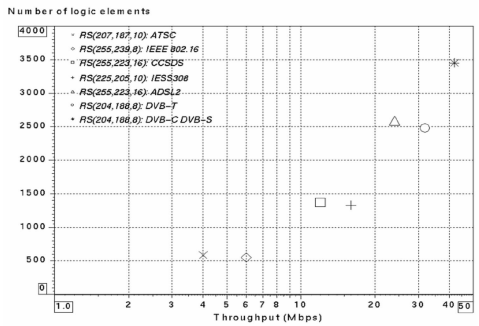

Figure 7 Logic size versus throughput different standards

In order to validate our methodology, we have implemented on the Sundance platform [6] the decoding part of a DVB-DSNG compliant receiver with a 26Mbits/s throughput constraint. Synchronization and interleaving parts have been implemented in software on the C6x DSP. For the interleaving part, a row writing / column reading process has been used with a 204x204 matrix.

Synthesis time results show that high level synthesis helps the designer in rapid prototyping. It allows fast implementations fulfilling various sets of system constraints (latency, throughput) and application constraints (number of coding states, …) from a single high-level description of the behavior.

4. Conclusion and perspectives

Traditional methods for rapid prototyping of hardware cores suffer from heavy limitations that prevent them from efficiently addressing both the algorithmic complexity and the high flexibility required by the various application profiles in fast implementation and prototyping issues. A high-level synthesis tool has been used for generating a set of hardware IP cores for Viterbi and Reed-Solomon decoders fitting various communication standards in a fastest way that hand coding. The generated architectures have been used in the case of a 26 Mbits/s DVB-DSNG decoder design. Results prove that high-level synthesis tools can manage complex algorithmic descriptions and generate high performance architectures. Moreover HLS allows fast prototyping for different behavioral parameters and architecture exploration. Further efforts in our rapid prototyping methodology including HLS will focus on exploiting the possibility on the integrator's side to parameterize the communication channels accurately in order to automate the generation of the communication interface.

5. References

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

[2] 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.

[3] G. Corre, E. Senn, N. Julien, E. Martin , *Memory Accesses management during High Level Synthesis*, CODES+ISSS, September 2004.

[4] P. Coussy, A. Baganne, E. Martin, *Communication and Timing Constraints Analysis for IP Design and Integration*, In Proc. of VLSI-SOC Conference, pp. 38-43, December, 2003.

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

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

[7] 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.