Design Automation in Digital Signal Processing: synthesizable VHDL models for rapid prototyping

Dequn Sun, Hans Peter Amann, Alexandre Heubi, Fausto Pellandini

Institute of Microtechnology, University of Neuchâtel
Rue de Tivoli 28, CH-2003 Neuchâtel, Switzerland
sun@imt.unine.ch

RÉSUMÉ

Cet article présente un exemple d'implémentation matérielle complète d'un filtre numérique en treillis RIF-RII. Des outils CAO ont été utilisés avec des modèles VHDL synthétisables optimisés pour des applications basse consommation du traitement numérique du signal. Sur la base de cette exemple, des comparaisons ont été effectuées entre la conception manuelle et une approche CAO moderne. Ce travail confirme les avantages de cette seconde approche qui reste indépendante de l'outil de CAO utilisé, de la plateforme informatique et de la technologie. De plus, l'accès à une implémentation matérielle très rapide est assuré même à des personnes peu expérimentées en conception VLSI. Des implémentations en différentes technologies pour ASICs et FPGAs ont été comparées.

ABSTRACT

This paper presents a complete hardware implementation example of a digital lattice FIR-IIR filter, using multiple CAD tools and basic synthesizable VHDL models optimised for low power consumption DSP applications. The example serves for comparisons between traditional full-custom design and modern, automated design schemes. The present paper confirms the advantages of the automated design approach that provides a very fast implementation method, design tool and platform portability, technology independence and hardware implementation access to people with limited knowledge in VLSI. As a secondary result, a comparison between implementations in several ASIC and FPGA technologies is provided as well.

1. Introduction

In modern design methodologies for electronics, top-down approaches using at many steps CAD tools, become more and more common. For several reasons (regularity, genericity, etc.), digital signal processing (DSP) is a field where this method can be brought to profit in a very efficient way.

Starting from high-level descriptions and passing through a resource-allocation and scheduling phase, a set of generic basic elements is selected, instantiated and interconnected. Using the corresponding descriptions in the form of executable models, e.g. written in IEEE's standard VHSIC Hardware Description Language VHDL, the subsequent hardware implementation steps down to the final ASIC or FPGA can be done automatically, and hence accelerated in an important way. The loose in efficiency – power consumption, Si area, max. clock frequency, etc. – can be corrected later, once the first implementation has proven the feasibility.

The aim of the present paper is (i) to show the pratical feasibility of the proposed design approach, and (ii) to compare hardware implementation results of a particular DSP architecture, called IMT-low power (*IMT-lp*), obtained by full-custom ASIC design and automatic generation of standard-cell/netlist ASICs and FPGAs. Comparison criteria are the design time, ease of propagation of modifications, consistency of documentation, power consumption, cost, silicon area, and, most important, the ease of use for (non-specialised) designers.

The paper is structured as follows. We first draw the design flow from top-level DSP specifications down to hardware (§2). Next, we introduce the IMT-lp architecture (§3) and the implementation example used: a lattice FIR-IIR digital filter used in a hearing aid application developed in-house, and present problems we encountered during the VHDL modelling phase (§4). The results are discussed (§5) and the conclusions drawn.

2. Design flow

Our design flow leads from high-level filter specifications to the final chip (figure 1). While the upper part is specific to our DSP application, the lower part has a more general signification.

2.1 DSP specific operations

A DSP application depends on four elements that are: (i) specifications, (ii) algorithm, (iii) target hardware architecture, and (iv) corresponding algorithm-to-architecture mapping rules. It can be shown that certain classes of DSP applications can be implemented with a relatively small set of generic hardware elements, and application dependency is limited to high-level parameters: (i) number of instanciations, (ii) parameter sets for basic building blocks, (iii) interconnections, and (iv) controller unit(s). Therefore, this kind of application is particularly well suited for an automated design flow.

As shown in the upper part of figure 1, DSP people working on algorithms and architectures can use their favourite tools to define the functionality and parameter sets of the generic basic building blocks. For individual applications, a resource allocation and scheduling scheme is used to determine the parameter sets mentioned above, while referring to the list of available resources.

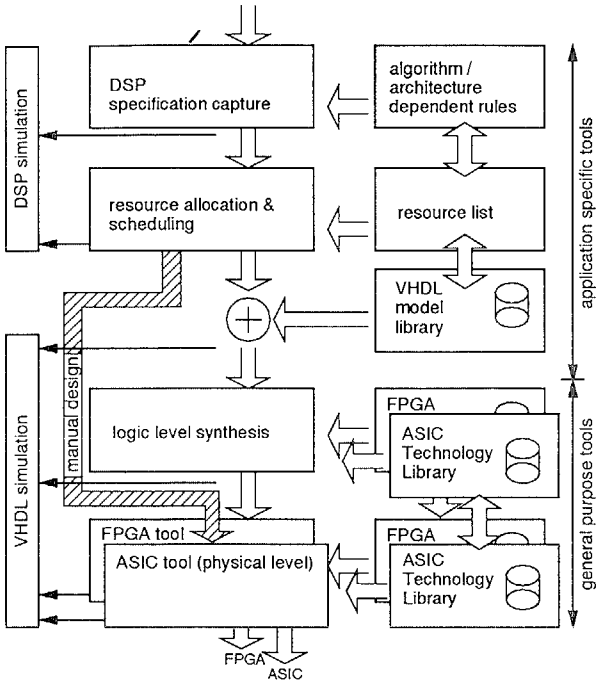

Figure 1: design flow for the DSP application

2.2 Implementation level

It can be seen that in the lower part of figure 1, the implementation operation can be performed according to the following two scenario:

I. Optimum Design: full custom manual ASIC layout, optimisation for low power, low area, etc. .

II. Fast Implementation: portable high level models, automatic logic level synthesis, standard cell implementation on ASICs and/or FPGAs, optimised for short turn-around time, fast implementation.

The choice depends on the design criteria, essentially on a balance between minimum design effort (high level of abstraction, functional description), degree of optimisation of the final circuit (cost, power consumption, area), the portability of the design, etc. . The first approach is well known, and the particularities of the architecture *IMT-lp* have been published before [1]. Therefore for the rest of the paper, we concentrate on the second scenario – *Fast Implementation* – and the comparison of the final results obtained with the two approaches.

2.3 Design flow for fast implementation

In a first step, a library of synthesizable VHDL models of the architecture's base elements has to be established. The results of the resource allocation and scheduling phase are then used to create a VHDL netlist instantiating these VHDL library models with their respective parameters, and to set up the ROMs. The information on control is also delivered through the resource allocation and scheduling phase and is used to generate the controller/sequencer units.

Data is then handed over to a logic-level synthesis tool where the desired functionality is mapped to base elements (standard cells) according to the desired target technology. As a result, a standard cell/netlist representation can be handed over to a (physical level) CAD tool, i.e. for place & route in ASICs or FPGAs. The results of both logical and physical synthesis can typically be extracted in the form of VHDL files, allowing hence a comparison between the original VHDL files and the VHDL files resulting from synthesis through simulation.

3. IMT low-power architecture

3.1 Architecture

IMT-lp is a novel architecture for a large class of digital signal processing applications (FIR, IIR, adaptive filtering, Fourier transformation, etc.), and has been optimised for a very low power consumption [1].

According to the application problem, the number of base elements and their combination can vary. Figure 2 shows the example of a FIR filter using this architecture.

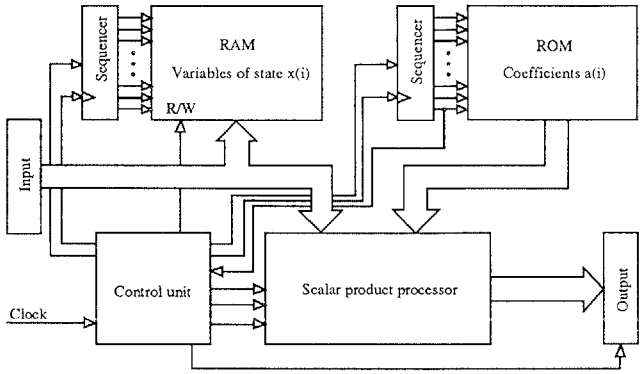

Figure 2: IMT-lp architecture in a FIR filter application

The scalar product processor (or multiplier-accumulator) is one of the most fundamental operators used in digital signal processing and has hence been optimised carefully [1].

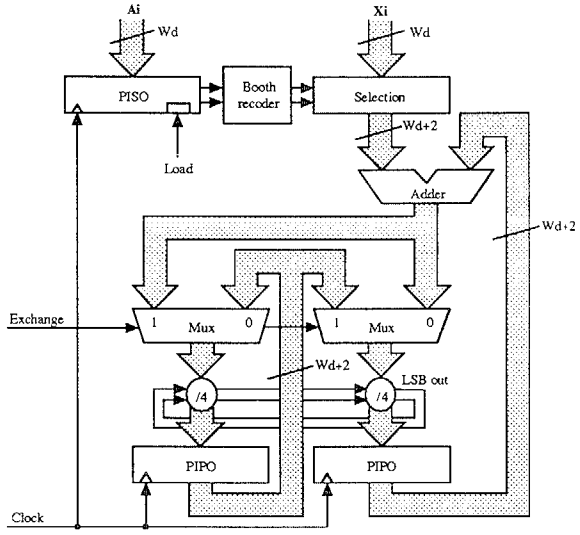

Figure 3: IMT-lp architecture: the scalar product processor

3.2 VHDL models

Modelling style

In order (i) to keep control over the synthesis process – important for DSP applications – and (ii) to reduce computation time, we decided for a modelling style using a very *structural* description. In order to keep the models portable, no product specific libraries have been used.

Particularities

ROM and RAM are usually generated with proprietary tools at the physical synthesis level. There are no standard memory cells in the technology libraries available for the tools we used for logic synthesis: we were therefore forced to build functionally equivalent blocks of relatively poor efficiency, one version for standard cell ASICs, a second one for FPGA, the former in the form of a matrix of 3-state buffers, the later as a standard behavioural model, both with the ROM contents in a local VHDL package. Similarly, the RAM model uses a

latch/3-state buffer matrix based structure for ASICs, and a behavioural description for FPGAs.

The *fan-out* of the models – while irrelevant for behavioural models – has to be considered for structural ones. We hence implemented an algorithm that partitions the building blocks and inserts buffers where needed.

4. Application example

A combined lattice FIR-IIR filter (16 bits, order 8 each) has been implemented successfully using our approach, both for standard cell ASICs and FPGAs. The synthesis process has been executed on SYNOPSYS' *Design Analyzer*, and the resulting EDIF netlist passed to physical level CAD tools: MENTOR GRAPHICS and COMPASS for ASICs, ALTERA for FPGAs. The functionality of the resulting hardware has been verified by means of post-synthesis gate level simulations with SYNOPSYS' VHDL Simulation System *VSS* and post-place&route simulations with COMPASS.

4.1 Exemple issue

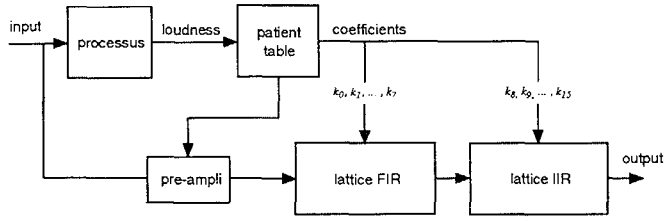

Fig. 4: lattice filters in a digital hearing aid

Figure 4 shows the block scheme of an all-digital hearing aid developed in our institute. After the pre-amplification, the digitised input signal is passed to the lattice filters for a frequency shaping operation. The set of filter coefficients k_i contains information on the patient's hearing defaults as well as on the predicted input signal intensity [2].

4.2 DSP specification for the lattice filter

For frequency shaping, a traditional lattice structure is used. Its high-level specification is illustrated in figure 5.

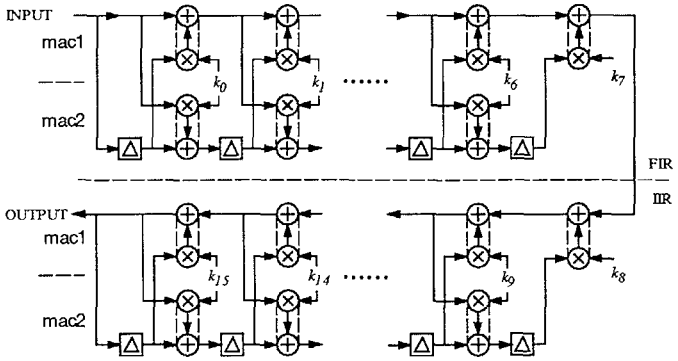

Fig. 5: data flow diagram of lattice FIR-IIR filter

It can be seen that a very small number of arithmetic operations is needed: multiplication, addition, and delay.

4.3 Resource allocation and scheduling

The IMT-lp architecture takes advantage of the regularity of the algorithm to simplify both scheduling and hardware implementation. Scheduling is organised hierarchically to limit the processing rate of each module to the strict minimum. Currently, resource allocation is done by hand [1].

Figure 6 shows the instanciated basic building blocks and their interconnections in the top-most architectural level of the lattice filter. It is composed of the following base units:

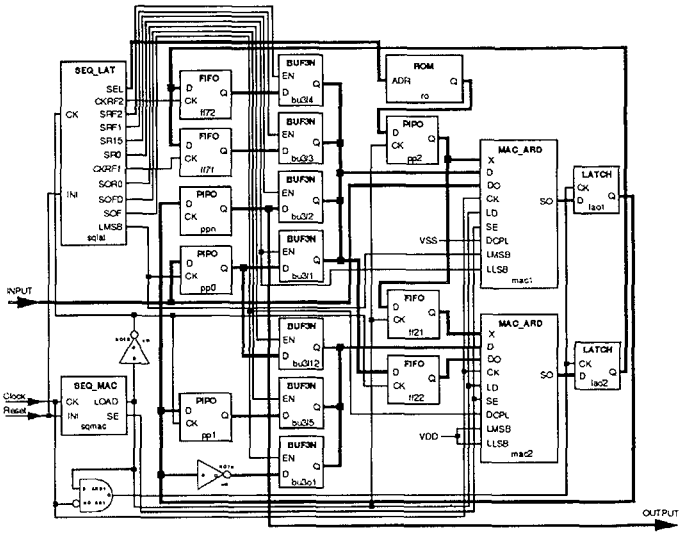

Fig. 6: instanciation and interconnection of basic building blocks after resource allocation and scheduling

MAC_ARD:	arithmetic unit (multiplier-accumulator including overflow and rounding control)
ROM:	local storage of the filter coefficients
FIFO:	first-in/first-out register for delay
PIPO:	parallel-in/parallel-out register
LATCH:	latch for temporary storage of the computation results
SEQ_MAC:	mac control unit (sequencer)
SEQ_LAT:	general control unit (sequencer)
BUF3N:	tri-state buffer array

4.4 FPGA implementation

For FPGAs, we concentrated on the feasibility test for a single implementation. Using ALTERA's FLEX8000 technology and MAXPLUS2 software, an FPGA of type EPF81188AQC208 with a device usage of 91% and a maximum clock rate of 10 MHz resulted.

4.5 ASIC implementation

The ASIC technologies we selected for the implementations are (i) *csel_lib* of CSEM (Swiss Center for Electronics and Microtechnique), (ii) *vsc370* of VTI (VLSI Technology Inc.), (iii) *ecpd10* of ES2 (European Silicon Structures), and (iv) *czb* of AMS (Austria Mikro Systeme International). In order to facilitate the comparisons between the different technologies and between full-custom design and automated standard cells/netlist methods, 1 µm CMOS processes have been used for all technologies tested.

4.5.1 CNM12 and CSEL_LIB (CSEM)

Our initial design has been done full-custom with this technology, and has been re-done using two more design methods for detailed comparisons:

Full Automatic (FA): automatic logic synthesis using VHDL models, automatic place & route.

Semi Automatic (SA): entry of the equivalent schematics using a COMPASS tool, manual mapping of standard cells of the selected target technology, automatic place & route.

Full Custom (FC): manually designed macro blocks at layout-level as well as some few elements of CSEL_LIB.

The results after place & route with COMPASS VLSI tools are shown in table 1.

	FA	SA	FC
chip size (mm^2)	3.84	2.13	2.0
nb. standard cells	2443	1305	217
macro blocks	-	-	4
nb. transistors	24144	12030	10998

Table 1: layout results

The functionality and the performance has been verified by means of post place&route simulations for all three implementation versions. The results are given in table 2.

	FA	SA	FC
supply voltage (V)	5	5	5
sampling frequency (kHz)	16	16	16
master clock rate (MHz)	2.3	2.3	1.3[1]
power consumption (mW)	4.3	3.7	2.4

Table 2: post-layout simulation results

A comparison of the methods FA and SA using both standard cells shows that the advantages of VHDL have been paid by a less efficient use of the silicon and a slightly higher power consumption for similar performance. Reasons for these differences might be (i) a non-optimal VHDL writing style, (ii) logic synthesis tools that can still be enhanced, (iii) standard cell libraries that are richer for proprietary standard cell placement tools than for logic synthesis tools and, finally, the influence of the users experience in the case of SA.

The maximum clock rates depend on several factors:

- delay distribution optimisation
- implementation technology
- quality of place&route tool for both FA and SA
- speed of the standard cells for FA and SA

Using a post-route simulation, we got a maximum clock rate of 20 MHz approximately for FC (equivalent to 40 MHz in FA and SA due to the difference in structure). For FA and SA – and with an optimised delay distribution – we estimated the maximum clock rate to 33 MHz before routing[2].

4.5.2 Other ASIC libraries

Technology independency of the automatic approach makes the implementation on different technologies very easy. Below, we give some interesting implementation results for different technologies (table 3).

	vsc370	ecpd10	czb	csel_lib
chip size (mm^2)	6.93	14.12	6.65	3.84
nb. standard cells	2439	3885	3004	2443
nb. transistors	29808	30428	28902	24144
max clock rate (MHz)	33	20	40	33[2]

Table 3: layout results with different ASIC libraries

The place&route operations have been accomplished with COMPASS' VLSI tool for vsc370, and with MENTOR's IC STATION for the others. The maximum clock rates have been obtained by post-synthesis simulation with SYNOPSYS' VSS. While these results correspond to the current version of our implementation, a more optimal distribution of the internal delays might lead to important enhancements, estimated roughly to max. clock rates two times higher.

5. Discussion

The differences between full-custom design and automatic synthesis based on HDL models can be resumed as follows:

	FA	SA	FC
technology dependency	very low	medium	high
tool dependency	very low	medium	high
fast protoyping	FPGA	?	-
development time: buildings blocks interconnect	 medium medium	 medium medium	 high medium
knowledge	VHDL medium	VLSI medium	VLSI expert
design modifications	easy	easy	difficult
design reuse	easy	easy, but tool related	difficult
know-how transfert	easy	tool dependent	very difficult
documentation	VHDL model	schematic	layout+ schematic
Si: cost / area / efficiency: FPGA ASIC	 expensive medium	 ? good	 - very good

Table 4: discussion

Considering the results of our experiences and the list of arguments here above, we are convinced that fast prototyping using high-level models and automatic synthesis tools are well suited to cover many of the implementation problems in DSP.

6. Conclusions

Considering the growing impact of modern top-down methods in general, and for DSP in particular, we presented the results of a comparison between full-custom ASICs and circuits obtained as a result of automated logical level synthesis (ASICs and FPGA), starting from high-level VHDL descriptions. While the method is very convincing for fast implementations – turn-around times from specification changes to prototype FPGA of half a day are possible – it is less appropriate for more demanding applications, where clock frequency, power consumption, silicon area, and price are more important than development time. Nevertheless, these methods will get more importance in the near future, taking profit from further enhancements in synthesis tools and richer standard cell libraries.

7. Acknowledgements

This work, funded by MICROSWISS and the Swiss National Found of Science, has been realised in collaboration with Sara Grassi, Michael Ansorge, and Peter Balsiger.

8. References

[1] A. Heubi et al., "A Low Power VLSI Architecture for Digital Signal Processing with an Application to Adaptive Algorithms for Digital Hearing Aids", Proc. EUSIPCO'94, Edinburgh, UK, Sept. 1994, pp. 1875-1878.

[2] S. Grassi et al., "A Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids", Proc. EUSIPCO'94, Edinburgh, UK, Sept. 1994, pp. 1661-1664.

[1] The difference in master clock rates of FC and SA/FA is due to a slight modification of the internal MAC structure, and should not significantely influence the results.

[2] The estimation is based on information on the technology as well as on the analysis of the critical path.