A LECTURE COURSE SERIES: FROM CONCEPT ENGINEERING TO IMPLEMENTATION OF SIGNAL PROCESSING ALGORITHMS WITH FPGAS

Mario Huemer, Michael Lunglmayr, and Markus Pfaff

Mario Huemer is with the University of Erlangen-Nuremberg, Institute of Electronics Engineering
Cauerstr. 9, D-91058 Erlangen, Germany
phone: +(49) 9131-85-27186, email: huemer@lte.e-technik.uni-erlangen.de
web: www.lte.e-technik.uni-erlangen.de
Michael Lunglmayr and Markus Pfaff are with the University of Applied Sciences of Upper Austria
Hauptstr. 117, A-4232 Hagenberg, Austria, phone: + (43) 7236-3888-2420
email: michael.lunglmayr@fh-hagenberg.at and markus.pfaff@fh-hagenberg.at respectively
web: www.fh-hagenberg.at

ABSTRACT

The design of complex modern embedded systems like wireless communication systems becomes increasingly inefficient. The methods used in the design process ranging from system concept design to hardware and/or software implementation are diverse, and typically managed by different teams with quite different expertise in a company. This naturally leads to communication problems between the teams. In this paper we highlight a new concept on hardware-oriented signal processing in education, where the same group of students passes through the whole design process. We describe a tripartite coordinated course series consisting of a lecture, a concept and simulation oriented exercise course using MATLAB and a practical course, where finally real hardware is developed using FPGAs (Field Programmable Gate Arrays).

1. INTRODUCTION

The design process leading from concept to realization, passes through three general levels of refinement, namely the algorithmic, the architectural, and the implementation levels. In companies, typically one team can be associated to one of these stages. The expertise of the teams typically diverge, furthermore the description of the system at the three stages of the design process is fundamentally different. Consequently the forward and backward communication between the teams turns out to be rather difficult.

One approach to solve these problems is to develop a consistent design flow providing unified design environments [1]. With our educational approach we additionally counter the communication problem between the teams by trying to account all of the three process stages within one complementary course program. A related pedagogical framework for teaching DSP hardware design can be found in [2]. Our program is made possible within the scope of the course of studies called Hardware/Software Systems Engineering: HSSE, where students are faced with hardware concepts, software concepts, and with system theory concepts already in the introductory study period.

After a description of the course parts and the applied development tools and boards, we highlight the results developed by a specific group of students during the concept development/simulation phase and during the hardware development phase. As an example we consider a chain of filters, decimation and interpolation blocks. The hardware is tested with audio signals on appropriate prototype boards. These boards provide a XILINX FPGA, audio codecs and appropriate interfaces. Of course audio signal processing is typically a signal processor application rather than an FPGA application, but as every student owns one of these prototype boards from previous courses, we decided to reuse the boards and to adapt the application accordingly.

2. ORGANISATION OF THE LECTURES AND PRACTICAL COURSES

The tripartite coordinated course series offered within the scope of the advanced study period consists of a lecture (2 SPPW: semester periods per week), a concept and simulation oriented exercise course (1 SPPW) using MATLAB intensively, and a practical course (2 SPPW), where finally real hardware is developed using FPGAs. It is assumed that all participants have basic knowledge on signal processing and on VHDL (Very high speed integrated circuit Hardware Description Language) programming. Since HSSE-students are already familiar with hardware development of datapath and control for microprocessors after the introductory study period they are able to fully concentrate on the specifics of signal processing hardware in this course.

2.1 Lecture: Implementation of signal processing algorithms

The lecture forms the basis for the exercise and the practical courses and has its main focus on fixed point effects and architectures for basic signal processing algorithms [3], [4]. Furthermore, concepts on minimizing the word length of samples and coefficients while minimizing the overall quantization noise are discussed. The lecture is structured as follows:

- Non-ideal effects in digital filtering
  - Floating point and fixed point numbers
  - Two's complement representation of numbers and the fractional format
  - Analog to digital conversion: Quantization noise
  - Quantization of filter coefficients
    - ∗ FIR (Finite Impulse Response) filters
    - ∗ IIR (Infinite Impulse Response) filters
    - ∗ Pole sensitivity
    - ∗ The cascade structure for IIR filters

- Quantized arithmetic
  - Rounding and truncation
  - Saturation and two's complement overflow
  - Limit cycles
  - Scaling of filter coefficients
  - Rounding noise
- Decimation and interpolation
- FFT (Fast Fourier Transform) implementations in fixed point arithmetic
- Digital signal generation in fixed point arithmetic
  - Impulse response generators for sine- and cosine signals
  - Direct digital synthesis (DDS)
  - Noise generators

For quantized arithmetic, a simulation library in MATLAB was developed, providing arithmetic operations for fixed point numbers.

2.2 The Exercise course using MATLAB: Investigation of algorithms and concepts in fixed point arithmetic

In this exercise course students produce basic MATLAB functions and scripts based on frameworks, which are provided by the instructor. Starting with simple examples on quantization noise introduced by analog to digital conversion, quantized filter coefficients in FIR- and IIR implementations and their effects on the original filter specification are analyzed. Improved structures like the cascade implementation for IIR filters are analyzed and implemented. Next the effects of quantized arithmetic are studied with 2^{nd} order IIR structures. Concepts like scaling of coefficients are implemented and tested in software. In the final exercise each team consisting of two members gets assigned one specific algorithmic problem, which has to be optimized in fixed point arithmetic using the functions and procedures generated in the latter exercises. A detailed specification has to be written which serves as input for the hardware implementation in the practical course.

2.3 The practical course: FPGA hardware implementation

For the implementation of signal processing algorithms, an FPGA rapid prototyping board called *Sandbox* is used. Fig.1 shows a picture of this prototyping board. The aim was not only to simulate the developed signal processing algorithms but also to demonstrate the processing of audio signals on such a development board. The prototype board provides a XILINX FPGA connected to a Texas Instruments audio codec via a I^2S interface for the purpose of this course.

The practical course already begins in parallel with the MATLAB exercises. It starts investigating simple signal processing hardware components which form the basis for the final filter implementation. For aritmethic calculations a special numerical package, similar to the library mentioned in section 2.1, which provides the data type *fractional* and basic arithmetic operations for this data type, was developed [5]. Both the data type and the arithmetic operations are synthesizeable in hardware. This *fractional*–format defines numbers in the range $[-1,1[$ with a fixed resolution of digits

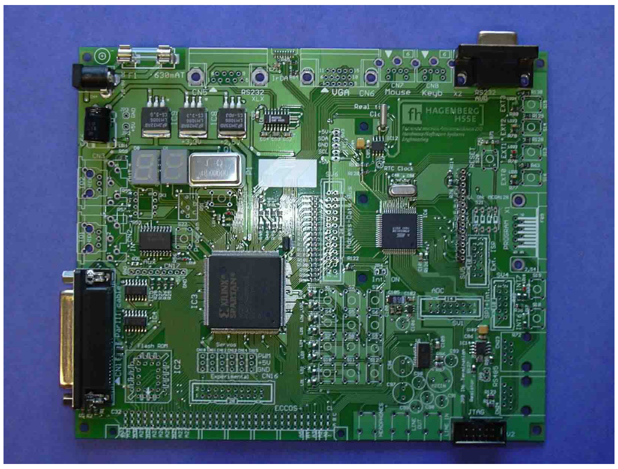

Figure 1: Prototyping board - *Sandbox*.

as follows:

$$x = a_0 \bullet a_1 \ldots a_{B-1} = -a_0 2^0 + \sum_{i=1}^{B-1} a_i 2^{-i} \quad a_i \in \{0,1\} \qquad (1)$$

As an important design principle, the data path/controller partitioning concept [6] is introduced. This concept allows to handle the complexity of advanced hardware designs and to build more efficient hardware in terms of area, clock speed and power consumption.

Components with the following functionalities are designed during this session:

- Signal multiplexing, serial/ parallel conversion
- FPGA interface for codec configuration and I^2S
- Multiply accumulate operation
- Signal generation
- IIR filter block
- Sample rate conversion: up and down sampling
- Shift registers for scaling

2.3.1 *Designflow and Simulation*

In todays hardware development process, the simulation of the designed hardware blocks is of great importance to find errors in early design phases. This keeps the development costs low. Because of this, a simulation environment for simulating these blocks is created using the *Modelsim*-Simulator from Mentor Graphics. This simulation environment allows reading and writing audio data from *.wav* files. This ensures that the developed signal processing algorithms can be tested on real audio data via this simulation environment without the time consuming synthesis step.

Because of the parallelism of the exercise and the practical courses, signal processing algorithms are developed according the design flow presented in Fig.2.

1. An algorithm is developed in MATLAB using double precision floating point arithmetic.
2. After that, this algorithm is ported to fixed point precision arithmetic for simulation in MATLAB.

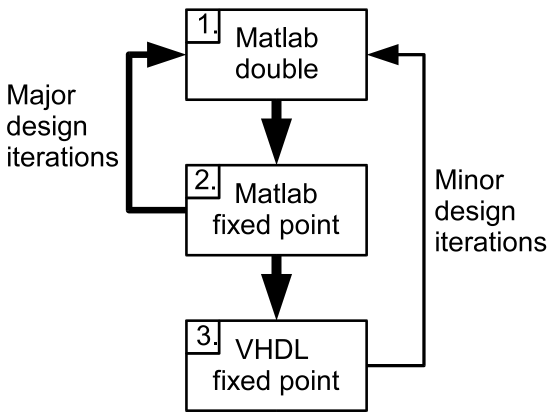

Figure 2: Designflow.

3. Next the algorithm is written in VHDL and simulated with the VHDL simulator using the simulation environment for reading/writing audio files.

Practice showed that design iterations between the VHDL hardware design, and the MATLAB simulation in double precision are very time-consuming. Because of the equal behavior of the fixed point arithmetic libraries in MATLAB and in VHDL, major design iterations should be done between step 1 and step 2 of the design flow. This helps to reduce the development time of the signal processing algorithms because design iterations between the double and the fixed point simulations in MATLAB are much less time-consuming than between the VHDL design and the MATLAB double realization.

3. AN EXAMPLE: FROM SPECIFICATION TO IMPLEMENTATION

After the completion of all the introductory and preliminary tasks in MATLAB and VHDL which are scheduled for the first half of the semester, each team consisting of two students gets a slightly varying specification for a band pass filter (see for example the specification in Fig.3).

This filter, which can be tested with audio signals should finally be realized on the *Sandbox*-FPGA.

3.1 Concept Engineering

The concept engineering tasks are scheduled for the last third of the exercise course which is entirely concentrated on the first half of the semester, so that the second half can be spent on the hardware design, the verification phase, and for possibly required design iterations. The concept engineering tasks are defined by the instructors as follows:

- Design of the filters in MATLAB using a cascaded structure according to Fig.4, consisting of two low pass filters, one high pass filter, one down sampling unit and one up sampling unit. This rather complex implementation structure has mainly been chosen for the purpose of practising different signal processing blocks rather than only implementing a straight forward filter structure without sampling rate conversion.
- Simulation of the filters in MATLAB using fixed point coefficients and fixed point arithmetic. The word length has to be optimized with the aim of minimizing chip area.
- Implementation of the required filters as IIR filters.
- Consideration of the limited capacity of the FPGA which mainly limits the synthesizeable number of multipliers for this application.
- Consideration of hardware reuse.

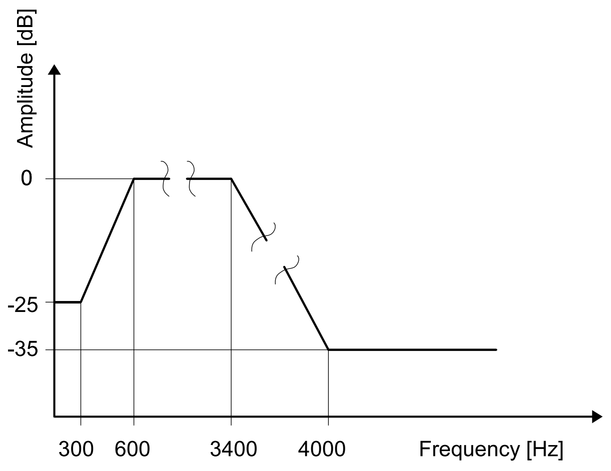

Figure 3: Example-specification of a band pass filter.

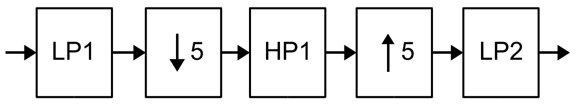

Figure 4: Cascaded structure of the band pass filter using three filters and two sample rate conversion units.

3.2 Implementation

To keep the design complexity low most implementations use one multiplier per IIR filter section. Although this is not very efficient in terms of area use, this implementation keeps the control logic of the design very simple. No superior control logic is needed, data is simply passed through from one hardware block to the next. Thus an extension of this structure can be achieved by simply inserting additional filter blocks. Fig.5 shows the structure of an example implementation of the filter structure. This implementation uses 4^{th} order IIR filters to fulfill the requirements.

To support scaling of the filter coefficients shifting units are included. The downsampling (most implementations used a downsampling factor of 5) is applied to decrease the relative bandwidth of the high pass filter. This is mainly done to minimize the pole sensitivity and to reduce the order of the high pass filter. The first low pass filter also acts as an antialiasing filter for the down sampling unit. Note that the

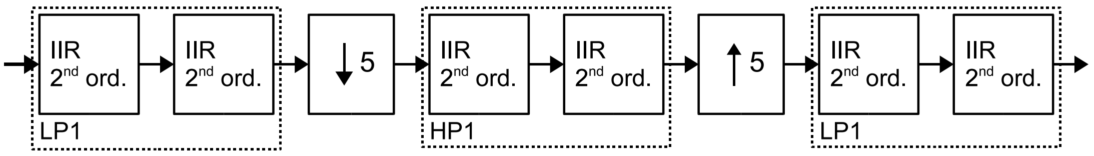

Figure 5: Implementation of the cascaded structure of the band pass filter.

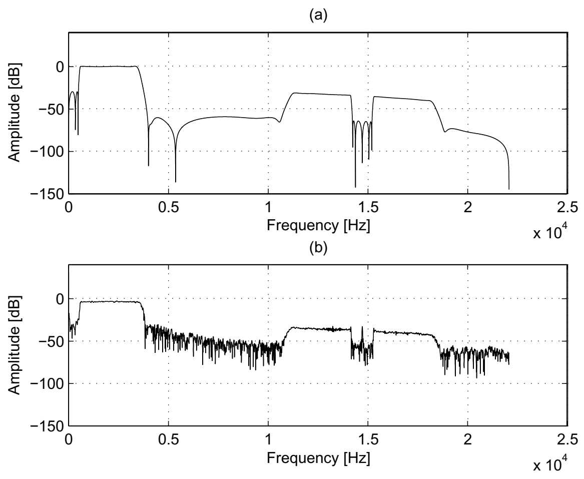

Figure 6: (a) Frequency response of the designed filter in MATLAB, (b) Frequency response of the fixed point implementation in Hardware.

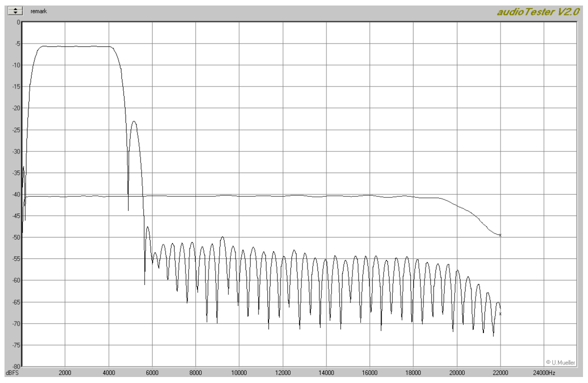

Figure 7: Measurement results of the frequency response of an FIR band pass filter realized on an FPGA.

sample rate conversion units also have shifting functionalities to support amplitude scaling effects when changing the sample rate [3].

3.3 Difficulties and Results

Because of the consideration of the fixed point characteristics during the whole development process, the amount of design iterations is kept low. Nevertheless, usually problems occur when porting the algorithms from the double precision format to fixed point. The typical critical points emphasized during the labs are the following :

- Problems with overflows and scaling
- Occurrence of limit cycles
- VHDL synthesis problems: Too large designs which do not fit into the FPGA

Beside these problems most students typically reach the goals within the scheduled time frames. Fig.6 shows an example of a satisfying implementation and the appropriate MATLAB results. Fig.7 shows the measurement results of a satisfying implementation of a band pass filter with a specification slightly deviating from the one described in Fig. 3. Here an FIR approach has been chosen instead.

4. CONCLUSION

In this paper we described a novel course series, which efficiently closes the gap between signal processing theory and signal processing hardware design in education. Traditionally courses focus on either signal processing theory or on digital hardware development. The latter rarely focus on signal processing hardware. Furthermore, these courses typically address different groups of students. With the new concept students undergo the whole signal processing development process, starting with theory finally ending up with optimized FPGA hardware.

REFERENCES

[1] M. Holzer, M. Belanovic, B. Knerr, and M. Rupp, "Design Methodology for Signal Processing in Wireless Systems," in *Proc. Mikroelektronik Informationstage 2003*, Vienna, Austria, 2003, pp. 303–307.

[2] T. Hall, D. Anderson, "From Algorithms to Gates: Developing a Pedagogical Framework for DSP Hardware Design", in *Proc.* 10th *Digital Signal Processing Workshop and* 2nd *Signal Processing Education Workshop*, Pine Mountain, Georgia, USA, 2002, pp. 157 – 161.

[3] D. von Gruenigen, *Digitale Signalverarbeitung (in German)*. Fachbuchverlag Leipzig, 2002.

[4] M. Werner, *Digitale Signalverarbeitung mit MATLAB (in German)*. Vieweg Verlag, 2003.

[5] W. Pauli, M. Pfaff, S. Reichoer, "DSP in Dedicated Hardware: Raising Value Abstraction for Fixed Point Implementation", On the *Conference CD-ROM of the International Symposium on Signals, Systems and Electronics (ISSSS' 04)*, Linz, Austria, August 2004.

[6] M. Zwolinski, *Digital System Design with VHDL*. Prentice Hall, 2000.