Unified UML Software Environment for Embedded Systems in Education

Bojan Kuljić*, Anita Sabo**, Tibor Szakáll***, Andor Sagi****

Subotica Tech, Subotica, Serbia
* bojan.kuljic@gmail.com    ** saboanita@gmail.com    *** szakall.tibor@gmail.com **** peva@vts.su.ac.rs

Abstract—This paper presents a software system which was designed as a tool for enhancing education process in the embedded system classes. Over the years it was established that students waste the majority of their time in classes on adopting software interface which leaves them with very little time to master the key principles in the curriculum. This is especially true in the case of the embedded systems because there students must learn to work with several software packets. The developed software system allows students to concentrate on just one graphical interface which is intended to handle every aspect of embedded system design – from algorithm simulation to the hardware implementation. Apart from this the system is capable of communicating with the third party software without burdening the user with the new interface.

Keywords: education, audio signals, filters, UML

I. INTRODUCTION

This paper describes the developed system whose primary use is to enhance the learning and skills of the students in the area of calculation and implementation of the DSP (Digital Signal Processing) solutions. In order to present the goals of this work more clearly, a functional schematic is given in Figure 1. as a demonstration of one such exercise. The main goal of the exercise was to demonstrate impact of the realized filter on the audio signal. The exercise was divided into five sections:

- calculating the filter on the basis of given parameters,
- testing the influence of the filter on the audio signal in the simulator (the audio signal was supplied in the form of a binary file),
- testing the influence of the filter on the audio signal with introduced noise in the simulator (mixed audio and voice signals were supplied in the form of a binary file),
- testing the influence of the filter on the audio signal on the real DSP platform (the audio signal was supplied through an audio generator which was attached to the audio input jack on the DSP board),
- testing the influence of the filter on the audio signal on the DSP hardware with addition of the noise (the audio signal was supplied through an audio generator which was attached to the audio input jack and the noise was supplied through a white noise generator which was attached to the second audio input jack; mixing of those two signals was performed inside the DSP processor after the A/D conversion).

Figure 1. Functional schematic of the system

II. CONCEPT

The main advantage of this concept lies in the fact that the students have to concentrate on just one customized programming tool and therefore there is no need to waste time on becoming acquainted with the multiple software packages, which leaves more time to concentrate on the material from the lectures. This way there is less pressure on the students and the professors time-wise which boosted the efficiency in the class and time is always of the essence because the students very often do not possess the background knowledge necessary to easily understand...
the study material. In order to enhance the educational process in the DSP area it was very important to achieve the ability to compare results of the simulation with the results obtained on the real hardware. That way it was possible to obtain the practical experience regarding the implementation of the theoretical calculations on the real DSP systems (e.g. to compare differences between the results obtained with 8 bit or 32 bit arithmetic in realization of the filter coefficients). This way the students could obtain deeper insights into the principles of the practical applications regarding the digital processing algorithms. Such observations were very difficult to achieve on just simulation software which operated with the 64 bit or even higher floating point arithmetic which made practically impossible to predict the influence of the reduced precision on the real fixed point DSP processors.

Likewise traditional DSP processors often have 16 bit registers with fixed point arithmetic so students could observe the direct influence of the rounding in the mathematical operations applied on the digital signal. The developed software supports the resolution decreasing. Even if the DSP hardware supports e.g. 16 bit operations, software could use just first 8 bits which creates the artificial 8 bit processor. This feature was intended to allow the students parallel view on the same filter realized in 8, 12, 16 and 32 bit resolution. This form of direct education enables the students to anticipate and practically verify behavior of the calculated filters on the real hardware whose precision in the mathematical operations was lower than the one used in the simulation software.

The purpose of the developed system was not to replace existing solutions but to allow the use of different software tools through one unified graphical environment. Its primary task was to generate the DSP algorithms and to display the results obtained with those algorithms in the simulation and real environments. The examination of the text in the exercise shows the methods applied in the educational process. It is obvious that there were many deviations from the conventional methods in the education in certain segments. In writing the technical literature the main guiding lines for authors are concision and precision in presented material with very little redundancy. The authors of this paper completely agree with those principles in subject presentation but only in the case where the audience was already acquainted with the basic knowledge in the presented material and the purpose of their interest in that literature was to broaden their existing knowledge. In the communication with the students it was concluded that features such as concision, non-redundancy and uniform representation in the presented material were the main obstacles in fast acceptance of the class material. The greatest problem the students were complaining about was the inability to recognize the important segments of the material which represented the basic principles of the subject. Without understanding those key principles the students could not proceed to the rest of the material. This meant that traditional methods of non-redundancy and concision gave the same amount of space in the study material even to the least important segments. This was even desirable when the reader had enough background knowledge about the subject in order to prevent wasting time in the reading but when it comes to the students it was determined that this method was almost always a great obstacle in the educational process. Besides this it was also concluded that it was necessary to include key terms and principles in the exercise, as often as possible, so that the students could more easily determine which parts of the material were crucial. In the previous figure it was shown that the simulation was performed in the Matlab while the Code Composer Studio was used for the hardware implementation. It is vital to mention that the end-user was not even aware of the existence of those software tools. Those tools were used in function of the internal modules and as such they could easily be replaced by other software solutions without awareness of the end user. For example, in order to switch to another hardware platform it was only required to change the cross compiler for that platform, also the simulation tool could be replaced with other tools like Octave. The Matlab was used as “black box” which received the input data, applied the mathematical operations and outputted processed numerical data. Basically, it was possible to write the graphical user interface in the Matlab for displaying data but there were two major obstacles:

- such graphical interface had to be first designed and then realized; students would have to be first trained to use developed solution before attending the laboratory exercise which would required a great deal of time,
- the tool would not address the DSP hardware but only the simulation, so additional time would have to be invested in the programming tools for the hardware (e.g. Code Composer Studio).

The above text clearly shows the need for a programming tool that would require very little time for introductory training and would be focused on the actual study material.

III. PRACTICAL REALISATION

Because of its interactive and graphical features this solution was proven to be a good candidate for application in the educational courses in the area of digital signal processing. In order to become fully acquainted with the software the students required only two hours of theoretical preparation and two hours in the computer laboratory.

The developed software allowed the students to more efficiently:

- understand the basic principles of FIR and IIR filters,
- understand the correlation between the filter parameters and real hardware,
- conduct practical experiments in the laboratory,
- develop the physical filters based on their theoretical models,
- enhance knowledge in the area of digital audio signal processing.

The DSP blocks were realized in the form of object-oriented classes. The end-user places the DSP block in the framework and makes relations between them in a form of a graphical arrays. Figure 2 presents the appearance of the framework in the case of the buffer realization through the dynamic array. Figure 3 shows an example of the framework in the case of the FIR filter development. The student introduces graphical blocks into the framework which in this case were the following:
Processing component (this component was developed for communication between other blocks and for gathering processed data),

FIR DSP block (it contains filter parameters),

DSK block (placing this block triggers activation of the cross compiler which communicates with the Code Composer Studio),

Simulator block (placing this block triggers activation of the cross compiler which communicates with the Matlab).

A double-click on the Processing component opens a pop-up window which was used for defining parameters of the FIR filter as shown in Figure 4. The final step was to compile project and display the compared results from the Matlab and DSP hardware as presented in Figure 5. The red line represents the results from the Matlab and blue line represents the results obtained from the hardware. Figure 6 presents the hardware setup used in the laboratory. On the right is the signal which is connected with the DSP hardware, in the middle, over the red wire. The output audio port from the DSP hardware is connected with the speakers through the blue wire. The DSP hardware is connected with the PC over serial cable.

IV. UML FRAMEWORK

The elements that formed the software were divided into two groups: the framework and the add-ons. In essence this means that the framework could be expanded with the units created to perform certain functions. One of the main tasks for the framework was to achieve independent operations between software and hardware in order to make it possible for different hardware platforms to be used without the influence on the user interface. This means that the developed framework implements the processing algorithm on the abstract level while cross compiler translates abstract form into physical realization for the DSP platform. The architecture of the framework was developed in object oriented UML language. The basic functionality of the framework could be expanded with the software add-ons.

Therefore the framework consists of only elementary modules which allows algorithm development and could be divided into two groups:

- units for data processing,
- units for data flow control.

![Figure 2. The buffer realization through the dynamic array](image)

![Figure 3. The framework in the case of the FIR filter development](image)

![Figure 4. FIR filter pop-up window](image)

The goal of the interactive development system in real-time was to perform validation of the development progress and functionality of the elements that were implemented on the targeted processor as well as comparing the measured results with the simulated results. This method was easily implemented in interactive development because of the high degree of validation and integration. The primary use of the developed system was the application in digital audio signal processing but it could be easily modified for other applications as e.g. writing technical documentation based on UML structure implemented in the software. Applications in the digital signal processing pose great computational challenges for the processor. As an example, one can calculate FIR (Finite Impulse Response) filter. Designing of the filter could be performed in the Matlab. Based on the calculated results the realized filter can be declared as valid. During the development system checks if the physical realization of the filter was correctly realized on the targeted processor and that filter behaves the same way as in the simulation. Since the Matlab and the DSP processors have different registers for storing filter coefficients compared results will always show deviations. But because of the possibility of comparison in the early phases of the development with this tool it was easy to make sure that the variations were always minimal. The modeling used in the system was very efficient for development and testing of the algorithms because projected UML diagrams were easily adopted for that use. Modifications performed on the UML diagrams were instantly viewable on the algorithms performance.

The developed method for interactive development in real-time has many advantages with application development in the digital processing area because it was possible to test algorithms in the early phases of the development.
This led to faster implementation because any errors were removed early on in the process. The application of this method was further augmented with the ability to communicate with existing software tools. Since this solution was based on graphical diagrams, the diagrams were divided into groups by function:

- units for software logic,
- processing units,
- peripheral units,
- control units.

There are many researches related to the realization of the development interface for digital audio signal processing which allows the modification of the parameters during the development. The basic set back of those solutions was the lack of ability to communicate with the existing programming tools mainly because their sole purpose was to simply demonstrate only one aspect in the development and were never meant to be used for integration with other solutions in order to achieve results enhancement. The DSL (Domain Specific Language) use has been in constant climbing in the last years but they are still not widely used in digital signal processing.

There were many attempts to develop the complete DSL language that could be used in the field of the DSP development and that could help the researchers regardless of their area of expertise. Even with the increasing application of the DSL language it has not become the standard in digital signal processing. According to this most energy and resources today are invested in the syntax and semantic development for the DSL language. Because of this most developed solutions are still based on the textual command interface which was designed to address only specific issues [1],[2].

This makes solving the simple tasks very easy but makes it rather difficult to develop the complex systems in an easy and intuitive manner. The existing DSL languages are mostly text-oriented and developed with object oriented methods. The main concern was to solve specific problems and not to define the basic building blocks which could be easily combined into the complex algorithms. Most of the existing tools use simulation as a verification method while the analysis of the results obtained from the hardware in real-time was neglected [3],[4].

The UML was chosen as suitable option for modeling the DSP blocks because:

- UML is recognized as the standard and it is widely used in the industry,
- UML possess the mechanisms for expansion (stereotypes, tagged values, advanced range verification etc.), which makes it possible to introduce the additional semantic into the existing language,
- UML is modeled through the meta-model mechanism which allows upgrading existing UML structure in the case of need,
- using the object oriented approach in the UML enables class definition through class interface and behavior class; this separation between the definition and the instance of the class allows the development of the libraries with reusable components (additional advantage lies in the ability to define new component by inheriting the features from the other components).

Furthermore the UML does not depend on some particular methodology so it is easily possible to define the specific methodology which fits the problem at hand. This has led to the widespread use of the UML modified languages for real-time and embedded systems. For instance, the HASOC methodology [5] expanded UML-RT in order to include a notation with mapped information’s.

V. CONCLUSION

Since there were no defined standards and regulations regarding the use of the modern electronic devices and the software tools in the educational process, the main contribution of this paper was the enhancement of the work with students in this domain. With this in mind a method was devised for working with students which included the development of the interactive graphical interface. The developed tool allowed easier interaction between students and the process of filter design. This work showed that the development of the DSL language based on the UML diagrams allowed achieving a high degree of interaction and efficiency.

REFERENCES
