Virtual Laboratory for Digital Signal Processing

Javier A. Guerra
Department of Electronic Technology
Universidad de Sevilla
Seville, Spain
jgcoronado@us.es

Samuel Dominguez-Cid
Department of Electronic Technology
Universidad de Sevilla
Seville, Spain
sdcid@us.es

Juan Ignacio Guerrero
Department of Electronic Technology
Universidad de Sevilla
Seville, Spain
juaguealo@us.es

Antonio Garcia
Department of Electronic Technology
Universidad de Sevilla
Seville, Spain
antgar@us.es

Diego Francisco Larios
Department of Electronic Technology
Universidad de Sevilla
Seville, Spain
dlarios@us.es

Carlos Leon
Department of Electronic Technology
Universidad de Sevilla
Seville, Spain
cleon@us.es

***Abstract*—Digital Signal Processor is a useful tool for learning and practice about filters for digital signals. The practices of the subject "*Procesado Digital de Señales*" made the students learn how to use it, and how to run some algorithms. But the hardware and laboratory restrictions and the complexity of subject provoked some availability problems of digital signal processor platforms. An effective solution is the creation of a virtual laboratory which connects to the real hardware, as explained in this document.**

Keywords— Virtual Laboratory, Digital Signal Processor, Algorithm, Practice.

I. Introduction

In the Electronic Engineering degree, taught at "*Escuela Politécnica Superior*", University of Sevilla (Spain), there is a subject in the last course called "*Procesado Digital de Señales*" (PDS). PDS is focus on learning general concepts about digital systems and signals and learning and apply some techniques in digital filter design. It also allows students to learn the theory of Digital Signal Processor (DSP). This theory is complemented by laboratory practices, where it is used a particular DSP for the implementation of these practices. The largest part of the subject is regarding signal transformation. In that, they learn about Discrete Fourier Transform (DFT) and some of its implementations, as Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT). Z transform and inverse Z transform is also learnt in courses.

Apart from theory lessons, students must do some practice lessons to pass the subject. In these practice lessons, they are engaged in implementing some filters and signal transformations, such as those mentioned above.

II. Problem issue

Having laboratories to do research and subject practices could be one of the best ways to get the students to improve their education for every self-respecting University. A compete laboratory allow students to put into practice theorical teaching and obtain their own point of view of the obtained knowledge, like what happens in the real use case. Unfortunately, it is impossible to enable all the hardware and software required for all the practices and research at the same time. Also, due to schedule and space reasons, it is complicated to obtain access to the laboratory at any given time. So, in case of isolated research or trying a single-handedly practice, they need to wait until there is some spare time in laboratory. For this reason, any alternative that allows using the laboratory hardware or software is always an advantage. In this case, what it is proposed is the use of a Virtual Laboratory (VL) as solution.

The idea behind this solution is the subject of extensive research. The most widespread alternative implies using specific software for modelling the needed components for the virtual laboratory [1] or even using mobile applications designed for this specific aim [2].

Regarding signal processing, there has been some interesting research in the last year. Some of them are referring to creating a virtual instrumentation laboratory [3] or using a mathematical approach regarding Fourier theory [4]. Nevertheless, all of them imply accessing the computer locally, which is the downside to overcome. In addition, the proposed architecture to solve the bottleneck in case of many petitions to the VL is an addition to other similar publications.

In this sense, we propose a general solution that can be accessible from any device with a web browser, with no need to stay in the same place that the computer that runs the VL software.

III. Hardware implementation

A. The C5505 DSP

TMDX5505EZDSP [5] is one of the DSP boards that it is proposed to use in practices. It is a device with 16-bit fixed point DSP development USB connection. A USB port provides enough power to allow an ultra-low-power C5505 processor to work, so an external power source is unnecessary.

This board allows a fast and easy evaluation of any C5505 processors. This tool has also an XDS510 emulator. This emulator is used to measure the source code debugging capacity and supports Code Composer Studio software.

Fig. 1. TMDX5505EZDSP board.

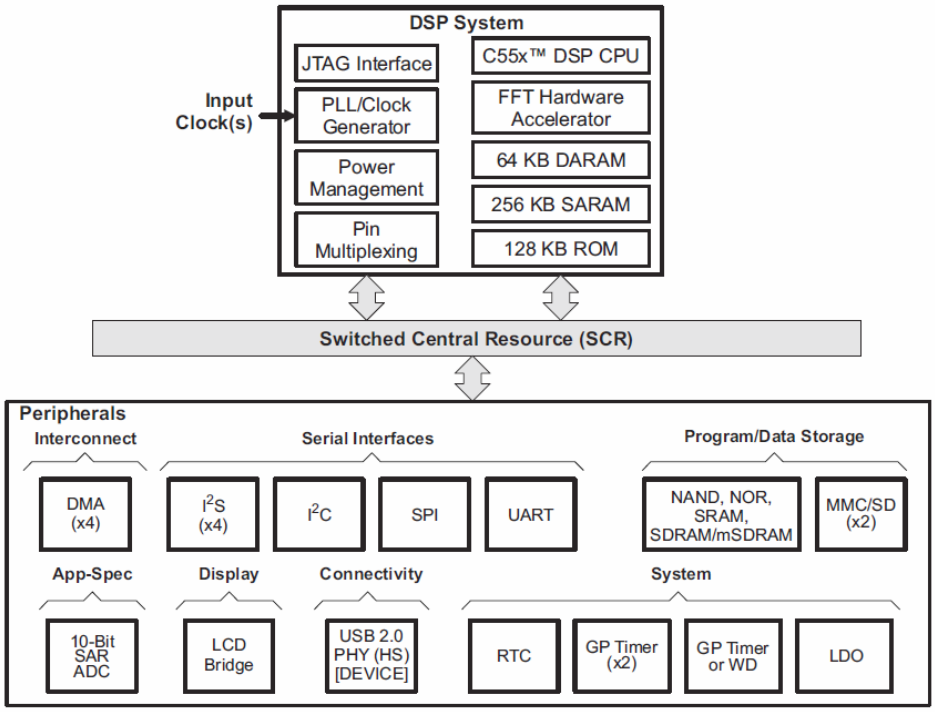

Fig. 2. TMS320C5505 USB Stick diagram [6].

TMS320C5505 is one of the lowest power 16-bit processor devices, but it can process 16 million of instructions per second (MIPS) and its onboard memory is up to 320 KB. All these specifications, and others like a Fast Fourier Transform (FFT) computer hardware accelerator, provides a higher integration than other comparable devices in the same price range. Therefore, C5505 is a base device for a large signal processing range application, such as voice recorders, musical instruments, portable medical solutions, or security applications. Therefore, we choose the TMDX5505EZDSP board. The diagram of the TMS320C5505 USB Stick is shown on Fig. 2.

Regarding peripherals, the TMS320C5505 has many serial interfaces, such as four integrated Integrated Interchip Sound (I^2S), an Inter-Integrated Circuit (I^2C), a Serial-Port Interface (SPI) and a Universal Asynchronous Receiver-Transmitter (UART). It also has four Direct Memory Access (DMA) controllers, and a Universal Serial Bus (USB) 2.0. All these connections make the device resourceful and suitable for this purpose.

B. *The C6713 DSK*

The TMS320C6713 [6] Digital Starter Kit (C6713 DSK) board, made by Texas Instruments, is the second board that is used in practices and is implemented in this VL. It is called a DSK because the C6713 DSP come on a board with some additional components, such as audio codec, several types of memory or peripherical control, as it is shown on Fig. 4. All these components made the DSK a great initiation and experimentation platform for signal processing applications development.

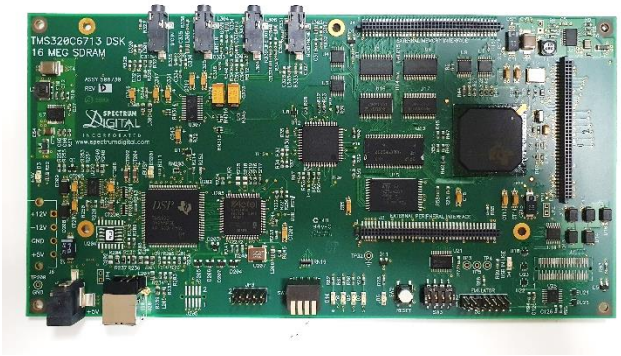

Fig. 3. TMS320C6713 DSK board.

The C6713 DSK board is connected to a personal computer (PC) with Windows as its operative system (OS) through the USB port. Thus, it is possible to load, deploy and debug the applications of the board. It has an AIC23 stereo codec to handle the analog inputs and outputs. This codec has a sample logic up to 96 kHz and a 32-bit resolution.

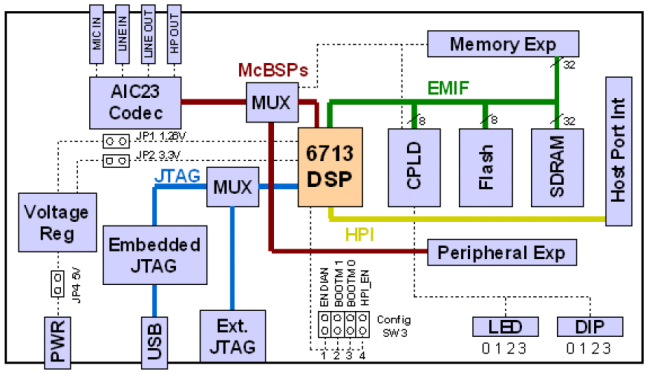

Fig. 4. TMS320C6713 block diagram.

The TMS320C6713 works at 225 MHz, which means up to 1800 MIPS and 1350 mega floating-point operation per second (MFLOPS). This DSP generation is built for high-precision applications, such as professional audio edition, medical applications or diagnostic and monitoring methods.

IV. Software Implementation

All these hardware devices need software prepared to send the specific code to the C550 DSP or the C6713 DSP (from now on, DSPs). In this case, the chosen software is Code Composer Studio (CCS). CCS [7] is an Integrated Development Environment (IDE) which provides a wide range of functionalities to help programming. It enables the design, implementation and testing phases, providing a C/C++ compiler, a source code editor, a programming project management environment, a debugger and many other characteristics for programmers.

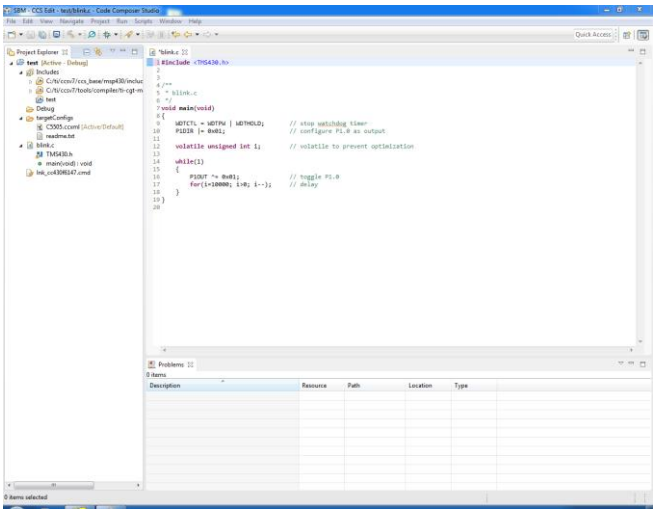

Fig. 5. Code Composer Studio.

CCS is based on Eclipse, another IDE. Because of this, it has the possibility of adding new packages, modules or plug-ins, which enable new functionalities to CCS.

Although there are newer versions than the one used in this paper, for this purpose the used version of this program is 4.0.

V. The Virtual Laboratory

This solution is made on a web-based platform, which could serve students and professors. "*Procesado Digital de Señales"* has a very important component of programming, and it could be complex for students without experience in programming. The Virtual Laboratory is proposed to make up for lack of experience in programming tasks. In this way, there is no need to access the laboratory. The computer is connected to use the resources that are provided by the DSPs.

Within the web-based platform, the authenticated user has an interface, called as Students' Virtual Laboratory (SVL), which is programmed to be as a middleware between the user and the DSPs command-line access. Because of that, the computer that has the C5505 DSP and the C6713 DSP running on it. The computer does not need any external peripherical, such as a monitor, a keyboard or a mouse. If the authenticated user is a professor, the Professors' Virtual Laboratory (PVL) interface is shown. The PVL provides tools to make the student tracking, including the students' doubts queries and responses.

A. *Students' Virtual Lab (SVL)*

The SVL has three main areas:

- The student identification.
- The practices configuration.
- The source code section.

In the student identification much information is included: name, last name, identification card number, and e-mail address are mandatory information, the students could optionally add a description, in which the students could propose doubts or observations. The practice configuration compounds: the practice number and target device.

The complete source code of each practice is provided in the laboratory session and in the Virtual Learning Platform of the University of Sevilla. Thus, the students know the files and places in which they must make a modification, and they do not need to completely write all the required code to run a DSP algorithm. Instead, they just need to specify the practice number and the target device, and the front end automatically inserts the code in the correct place into the source code file.

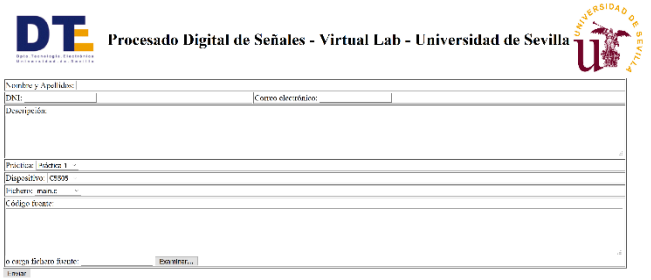

Fig. 6. Students' Virtual Laboratory front-end.

To avoid unwanted access to critical part of the code that could allow an execution of malicious code, the student only has access to write in certain part of the code, indicated by the number of selected practice and the target device. The user cannot write in a forbidden part of the source code, because the front-end controls the insert place and identifies the infinite loops by checking the loop condition.

The code writing area does not have to be just one. In fact, several practices need the code to be written in more than one place in order to run the algorithm. Hence, the SVL can add one or more text areas and selection controls, to select the target of the text area below, with the specific parts of the code. Thus, in these areas the user can write to run the algorithm and overcome the practice. There is also the possibility to upload a text file with the source code.

B. *Professors' Virtual Lab (PVL)*

The PVL has several sections:

- General statistics.
- Tracking of practices.
- Detailed tracking of students.

Fig. 7. Professors' Virtual Laboratory front-end, general statistics.

The general statistics (Fig. 7) shows the information about the number of connections, periods of operation, performed practices, pending practices, number of students, and average number of tries by student.

The tracking of practices (Fig. 8) shows information about the different tries in each practice, how many practices have been successfully run and how many have been unsuccessfully run by infinite loops, abnormal termination or incorrect results.

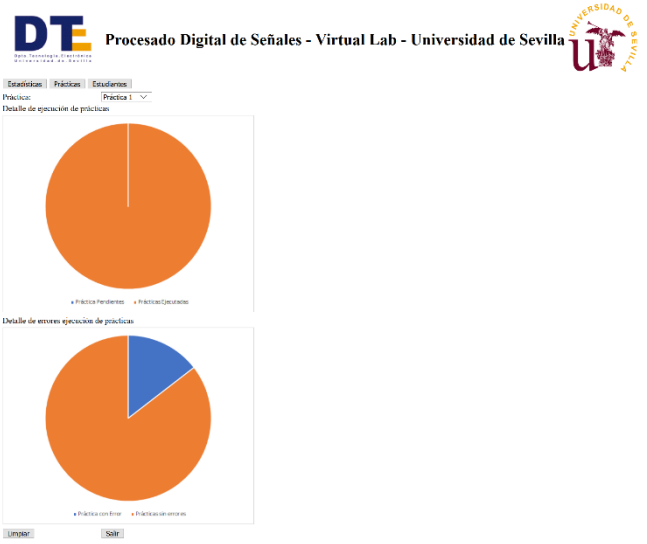

Fig. 8. Professors' Virtual Laboratory front-end, tracking of practices.

The detailed tracking of students (Fig. 9) allows to check and deep into advances of each student, showing the source code of different tries, results, and the possible doubts provided in the general description text area. The professor could response to these doubts directly using the own interface. This response is sent by e-mail to the student.

Fig. 9. Professors' Virtual Laboratory front-end, tracking of students.

VI. The SVL Architecture

Once the Virtual Laboratory has been described, we proceed to describe the complete process from an overall perspective, as is shown on Fig. 10.

The workflows are as follows: a student uses their personal computer (PC) to access to the website the SVL is hosted. This includes the use of some input/output peripherals like keyboard and mouse. The connection between the PC of the student and the remote Virtual Laboratory is through Hypertext Transfer Protocol (HTTP) to a specific domain.

The abovementioned remote SVL has been logically divided into two parts, apart from the web front-end where the student writes their algorithms. The first part consists of a queue manager (QM), whose work as an intermediate layer that filters the incoming requests to the remote PC that has the DSP running on it.

The QM oversees all incoming requests, manages them, and transfers them to the DSP program only when is completely available to run them. Thereby, the QM protects the system from any possible overflow of incoming requests from the students.

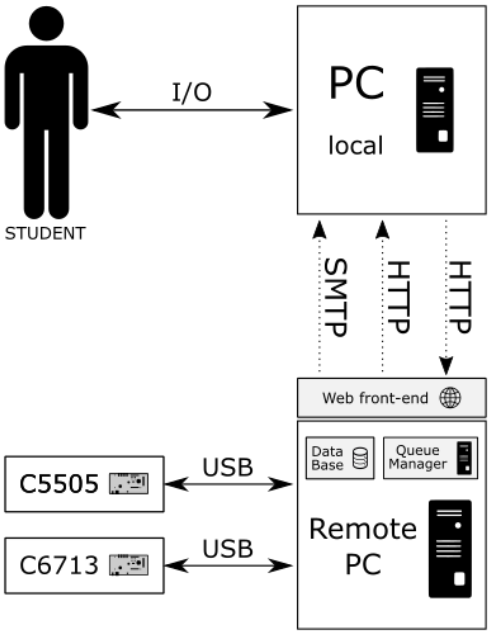

Fig. 10. Virtual Laboratory schema overview.

The remote PC accepts the algorithm sent by the QM to be processed. This PC has both TMS320C6713 DSK and TMDX5505EZDSP boards connected by USB. Thus, the run program, which has a command-line interface, loads the algorithm in the appropriate DSP. Only one of those DSP could be in running mode in each try. The created architecture cannot run different algorithms in both DSP in the same iteration.

The chosen DSP executes the selected program and generates a result. This result is sent to the student, so it goes the other way around, until it shows up in one of two ways: by email or by the front-end, using for them the protocols Simple Mail Transfer Protocol (SMTP) or HTTP respectively.

In case the students chose to receive the code from the execution of their code by email, it is necessary that they first provide the email, whether it is the college intern or their own. This option may not be as intuitive as receiving the code directly in the main web page, but it makes sure that the response will be returned and accessible for the student in their mailbox, although the response would take some time to be sent.

On the other hand, it is possible to receive the response directly on the front-end web page. If the student chose that option, in most cases it will be faster than the email option, but it is possible to take a long time to show the solution. It is due to the QM and the amount of the requests it must process. If the stack of requests in the QM is too full, the DSP needs some time to process each request, and it is not possible to estimate the elapsed time to execute the desired request and return it to the student.

Additionally, the QM manages the information of practices sent by students. Thus, the QM oversees recover after server breakdown. At the same time, the information of all practices is stored in a database, based on MySQL.

In both cases, the student receives the raw response to the execution of its algorithm, which fits with any of the proposed practices abovementioned.

VII. The PVL architecture

Apart from the previous perspective, the PVL has a different schema for the professor (Fig. 11) In this hidden schema, the professor has access to all the logs, which have been written each time a student uses the SVL to test their algorithms. For this purpose, the remote PC has its own specific access, separate from the SVL normal access.

Thus, the professor has complete control of every single line of code the students write. This allows the teacher to follow the practices of each student in a personalized way. In this case, it is necessary to grant full access to the remote PC, so it is done by a secure shell (SSH) access.

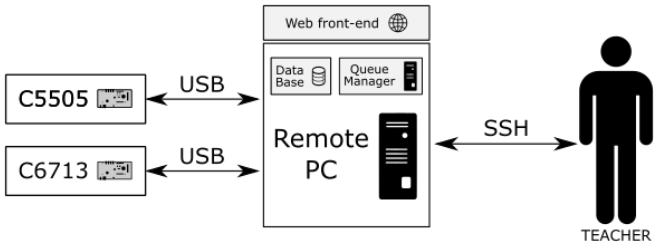

Fig. 11. PVL architecture for the Virtual Laboratory

The professor does not connect through QM. However, the professor could modify the queue constraints and order, although this option is provided by the command line. In this command line, the professor could remove practices which could jam up the QM or, even, the CCS.

VIII. Practices scheduling

The laboratory contents are structured into seven practices, which they could be run in both devices (TMS320C6713 DSK and TMDX5505EZDSP):

- Practice 1. Introduction to real-time digital signal processing. This practice is about DSP architecture, identification of different parts of DSP device, and introducing the CSS Interface Development Environment (IDE).
- Practice 2. Implementation and fundamentals of DSP.
- Practice 3. Design and implementation of FIR filters.
- Practice 4. Design and implementation of IIR filters.
- Practice 5. Analysis on frequency domain and the discrete Fourier transform.
- Practice 6. Digital Image Processing.
- Practice 7. A real application of DSP. According to the student's identification number, the student would have to develop an application related to: adaptive filtering, digital signal detection, digital signal generation, adaptive echo cancellation, speech signal processing, audio signal processing, and digital image processing.

The contents and wording of practices are provided by means of Virtual Learning Platform of University of Seville, which is only available for students enrolled in PDS.

Additionally, the wording of practices is complemented by Matlab and GNU Octave implementation, in order to provide a simulated and visual approach to introduce the concepts of the practice for students. Matlab and GNU Octave provides different tools to analyze the algorithm and results, which could improve the learning process of students.

IX. Results

Once implemented the VL, some functional test have been carried out, obtaining the following average execution times per student:

- The load of the algorithm in the program takes about 30 seconds.
- The minimal execution and return of information, which is in case of an error, takes about 90 seconds.
- In case of success in execution, the execution time depends of the practice been carried out. This time takes from about 10 minutes in the shortest practice, to about 26 minutes and 50 seconds in the case of the longest practice, because of the analysis of a 10-minute audio track during execution.

In summary, these runtimes via VL are close to the runtimes of the corresponding computer in the laboratory, which time gap of less than a minute in all cases.

X. Conclusions

The present paper provides a tool to remotely run practices of PDS. The SVL front-end allows students to run their practices as they need. The PVL front-end allows professors to make the following and reviewing of the students' practices. Additionally, the student receives the information and results about the practice by means of e-mail, having the possibility to propose doubts and commentaries.

Thus, the proposed tool is a very useful tool not only for the student, but for professors too. Additionally, it provides access to the technology used in the practices, which could be expensive for a student. Because of the way this VL has been made, it is possible to export to for use in other institutions. In this case, it is needed to load the application into the server and storing the front-end into the domain of the institution.

In the current scenario, in which the world is dealing with a sanitary emergency (provoked by an epidemic virus) or in the case of online education empowerment, it could be a good support to continue with the education of students in case of quarantine of professors or students.

XI. Future works

The future works is centered on:

- The generalization of front-end to add new practices and optional contents.
- Add front-end for remote administration of QM, CCS, and MySQL. Currently, this is performed by means command-line interface.
- Makes different analytics to provide advanced results in the practices and following of the students' evolution.
- Add a tool to online assistance for students when they are writing the source code, detecting infinite loops, mistakes, unknown instruction or names, etc.
- Add additional authentication process, including Lightweight Directory Access Protocol (LDAP) and a link with Virtual Learning Platform of University of Seville.

XII. References

[1] A. A. Sutchenkov and A. I. Tikhonov, "Electrical Engineering Materials Virtual Laboratory," in *2018 IV International Conference on Information Technologies in Engineering Education (Inforino)*, Oct. 2018, pp. 1–4, doi: 10.1109/INFORINO.2018.8581843.

[2] W. Zheng, L. Feng, B. Liu, P. Fu, and J. Qiao, "Development of virtual laboratory application structure in Android cellphone for distance learning," in *2017 First International Conference on Electronics Instrumentation Information Systems (EIIS)*, Jun. 2017, pp. 1–5, doi: 10.1109/EIIS.2017.8298575.

[3] R. Arsinte, T. Sumalan, and E. Lupu, "On the use of Virtual Instrumentation concepts in the test of Embedded signal processing applications," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, Jul. 2017, pp. 1–4, doi: 10.1109/ISSCS.2017.8034916.

[4] K. H. Cheong and J. M. Koh, "Integrated Virtual Laboratory in Engineering Mathematics Education: Fourier Theory," *IEEE Access*, vol. 6, pp. 58231–58243, 2018, doi: 10.1109/ACCESS.2018.2873815.

[5] Texas Instruments, "TMS320C5505 Fixed-Point Digital Signal Processor Datasheet." Sep. 2013, Accessed: Jan. 09, 2020. [Online].

[6] R. Gummattira, P. Baltz, and N. Seshan, "TMS320C6713 Digital Signal Processor Optimized for High Performance Multichannel Audio Systems," p. 12.

[7] "CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) | TI.com." http://www.ti.com/tool/CCSTUDIO.