Int. J. Adv. Eng. Pure Sci. 2021, ASYU 2020 Special Issue: 78-87
DOI: 10.7240/jeps.897556

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

Low-Latency SoC Design with High-Level Accelerators Specific to Sound Effects

Ses Efektlerine Özel Yüksek Seviye Hızlandırıcılarla Düşük Gecikmeli SoC Tasarımı

Yunus Emre ESEN[1], **İsmail SAN**[1]

[1] *Eskişehir Technical University, Department of Electrical and Electronics Engineering, 26555, Eskişehir, Türkiye*

Abstract
High-quality sound processing requires hardware acceleration in order to reduce the processing latency of the applied sound effects. Computational latency of producing enhanced sound from the audio input is an important delay component and affects the performance especially in artists' live performance or high-quality sound generation. Artists want to apply a sound effect in real-time on their music and latency is the main problem when these systems running in real-time. CPU-based systems present flexibility, but introduce a high amount of latency while processing, which in fact affects the artist negatively. In this study, to get the flexibility through software and the acceleration via hardware specialization, we present a system-on-chip (SoC) solution with HW/SW co-design methodology for some sound-effects. We reduce the latency and increase the frequency by applying pipelining through MATLAB. The system is implemented and tested on a programmable SoC platform, ZedBoard, which contains ZC7020 Zynq chip with a dual-core ARM-Cortex-A9 processor and a reconfigurable FPGA part. The ARM processor enables the management of sound-effect hardware accelerator running on FPGA and provides communication with user. A sound effect is designed with block models provided by MATLAB & Simulink at high-level. MATLAB HDL Coder then converts these blocks into RTL-level hardware designs. The followed design methodology provided by MATLAB & Simulink enables high-level block design that can be embedded into FPGA at RTL-level to benefit from the speed provided by high-speed hardware registers and to have an AXI interconnect interfacing with software in order to utilize the software flexibility. The study shows that latency is reduced significantly.
Keywords: hardware accelerator; system-on-chip; pipelining; sound effect.

Öz
Yüksek kaliteli ses işleme, ses üzerinde uygulanan efektin sebep olduğu işleme gecikmesini düşürmek için donanım hızlandırması gerektirir. Ses girişinden gelişmiş ses oluşurken hesaplanan gecikme, önemli bir gecikme bileşenidir ve özellikle sanatçıların canlı performansında veya yüksek kaliteli ses üretiminde performansı etkiler. Sanatçılar, müziklerine gerçek zamanlı bir ses efekti eklemek ister ve bu efektler gerçek zamanlı sistemlerde kullanıldığında gecikme ana problem haline gelir. CPU tabanlı sistemler esneklik sunar, ama işleme esnasında oluşan büyük miktardaki gecikme sanatçıları olumsuz olarak etkiler. Bu çalışmada, esnekliğin yazılımla ve hızlandırmanın donanım özelleştirmesi ile elde edildiği, bazı ses efektleri için yazılım/donanım ortak tasarım yöntemi ile bir sistem üzerinde çip (SoC) çözümü sunuyoruz. MATLAB üzerinden ardışık düzen uygulayarak gecikmeyi azaltıyor ve frekansı artırıyoruz. Sistem, çift çekirdekli ARM Cortex A9 işlemcisi ve yeniden yapılandırılabilir FPGA'e sahip ZC7020 Zynq çipi barındıran programlanabilir bir SoC platformu olan ZedBoard'da çalıştırılmış ve test edilmiştir. ARM işlemcisi, FPGA üzerinde çalıştırılan ses efekti donanım hızlandırıcısının yönetilmesine ve kullanıcı ile haberleşilmesine olanak sağlar. Bir ses efekti MATLAB & Simulink tarafından sağlanan yüksek seviyede blok modeller ile tasarlanmıştır. MATLAB HDL Coder, bu blokları RTL seviyesinde donanım tasarımlarına dönüştürür. MATLAB & Simulink tarafından sağlanan bu tasarım yöntemi, yüksek hızlı donanım yazmaçlarının sağladığı hızdan yararlanmak için FPGA'ya RTL seviyesinde gömülebilen yüksek seviyeli blok tasarımı yapılmasına ve yazılım esnekliğinden yararlanmak için AXI ara bağlantısı ile arayüz oluşturulmasına olanak verir. Bu çalışma gecikmenin önemli bir ölçüde düşürüldüğünü gösterir.
Anahtar Kelimeler: donanım hızlandırıcısı; çip üzerinde sistem; ardışık düzen; ses efekti.

I. INTRODUCTION
Digital signal processing (DSP) is a key technology in many signal processing applications such as speech and image compression, audio filtering, classification and coding where processing operations are performed in digital domain. Many real-life examples which are possible with DSP technology have already been in our daily life such as video streaming, computer applications, audio effects etc. In real life, signals are in continuous form such as a sound signal that is heard. In order to process a signal in a conventional computer, continuous signal must be converted to digital format that computer can understand and perform the processing. First, real-world signals such as temperature, pressure, audio, or video are transformed into digital domain, then signal processing algorithms run on digitized signal samples and manipulate the digital signal data to get the desired transform.

Corresponding Author: Yunus Emre ESEN, Tel: +90 554 390 91 97, e-posta: yunus_emre_esen@eskisehir.edu.tr
Submitted: 15.03.2021, **Revised:** 20.11.2021, **Accepted:** 20.11.2021

Finally, the digitized data needs to be transformed back to real-world continuous signal again [1]. The most important problem here is relatively long processing delays. There are many delay components in a processor-based system. Some applications can tolerate long delays; however, some delay-intolerant applications require a special attention in order to reduce the overall delay introduced by computation or communication. Hence, specialized hardware architecture with DSP-enabled components that is tailored to the signal processing algorithm is a powerful solution.

The demand of people on the low-latency applications is increasing gradually in every year. Especially in last 30 years, with evolution of technology, microprocessors and specialized hardware are used and developed with newer techniques in order to obtain better throughputs and better execution times. Some of the solutions are being developed with high-level software programming language for CPU-based solutions, which gives more design productivity and development opportunities, but incurs a cost of processing time for application being developed [2]. Application-specific hardware design provides very low latency for the selected application and gives better processing times with very high-throughput values (Ahmed Elhossini, Shawki Areibi, Robert Dony, 2006), but this costs the designer to lose design flexibility that comes from higher level programming languages. This article presents a case study to use the advantages of the two mentioned design flows in a system-on-chip (SoC) design with a HW/SW codesign methodology: (1) designing a DSP block with a high-level programming tool in MATLAB & Simulink, (2) converting this application-specific design to a specialized hardware with MATLAB HDL Coder, (3) managing and controlling the generated this special hardware design from a software application running on ARM-based processor and (4) demonstrating how pipelining can be implemented to DSP designs to achieve higher frequency and how much its effect on latency times compared to our previous solution LowLAG [4].

Many works have been carried out in the past years for sound effect applications. Some of these have been carried out only on the software platform [5] and some have only been carried out on the hardware platform [6]. On the DSP side, main concentration of this article is to cover to audio applications, especially in real-time sound effects. Sound effects are latency-intolerant application for real-time applications. For instance, a person speaking to a microphone and listening the voice with headphones can easily notice a latency if the latency is greater than 5 milliseconds [7]. Software applications offer latency values on these boundaries. Designing complex software applications with high CPU loads or running an application that requires high performance on the CPU can reduce the performance of the software. For such cases, software solution does not provide enough performance. High performance or high energy efficiency in computing can be achieved with compute and communication specialization for a specific task. Reconfigurable computing platforms allow us to tailor our design to perform better in terms of performance and area. In hardware, arithmetic and memory access of an application can be defined in spatially if the application has some parallelism. Spatial execution defined in hardware brings more performance- or energy-efficiency compared to serial execution model compiled into CPU. Because software applications are being developed with a high level language (such as C/C++), and even if the computation is described in one line of code that makes one useful operation at high-level, it may correspond to many lines of assembly language code. Among the instructions compiled from one-line of code, there are many instructions required to feed the data to the execution unit and it takes several cycles. However, the same operation can be described more efficiently in hardware. This gives the main advantage of low latency introduced by specialized hardware designs compared to software designs. As the software part of the SoC design handles the low load required processes such as address definitions, providing parameter inputs, controlling the hardware units etc., while hardware handles high load required processes. This HW/SW co-design methodology brings advantages of both software and hardware together. There are several studies on sound effects applications that are implemented on system on chip platform such as [8] [9] [10].

In this work, we developed our first work, LowLAG [4], while following the same methodology. There are three main contributions compared to first design, which are: obtaining better latency values with pipelined design (1), controlling hardware specification from software side (2), and designing new sound effects on Simulink that are based on a filter created via FDA Filter Designer Tool.

II. HARDWARE/SOFTWARE CO-DESIGN METHODOLOGY FOR LOW-LATENCY AUDIO PROCESSING

2.1. Low-Latency Sound Effect Design with Specialized Hardware

One of the competition points among Today's Tech companies is decreasing the latency and producing real-time products. Music is one of the important area since there are many real-time products available today. All techniques for low-latency and high-performance are utilized in the design stage (producer's side) in order for a person who listens to the music to get the best result. The artists, musicians and producer want to hear whatever they have done immediately. Because music is all about harmony, and even a single millisecond delay can disturb the whole harmony and cause decreasing the creativity. This is another motivation point that highlights the need for acceleration on sound effects in terms of low-latency.

Software solutions have limited processing capability, which make the specialized hardware design inevitable in low-latency targets. But, designing a specialized hardware is difficult in terms of design cost, long development cycles, and huge verification efforts. We propose a sound effect design with high level system blocks on Simulink, test and simulate on a personal computer, convert the MATLAB & Simulink design to a hardware design in VHDL language generated via MATLAB HDL Coder, integrate the design to a SoC architecture containing ARM processor and implement the whole SoC in ZedBoard programmable SoC platform. Generated RTL-level specialized hardware design can be integrated into a SoC architecture as an IP (Intelligent Property) core, which can be explained briefly as a reusable part of hardware system in an FPGA (Field-Programmable Gate Array) based SoC system [11].

The desired solution for low-latency audio processing in our design requires processing the audio sample by sample which is synchronized with system clock. Generally, 44.1 KHz is used as a sample rate for most of the audio files, and we designed the system for this sample rate. In ideal conditions, an audio file that has two channels (right and left) have 88200 sample for one second. Our methodology brings the necessity to run at least 88.2 MHz frequency for system core clock and supporting this frequency within the IP core. Pipelining in the IP core design provides to reach especially this frequency requirements rather complicated effect designs.

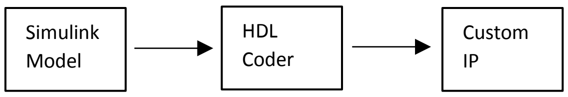


Figure 1. Simulink model to custom IP core

There are two different pipelining methods provided by HDL Coder that are implemented in this work. First method is adding new delay blocks to design stage of the Simulink model and using these blocks as registers. There is a special parameter to determine how many cycles are going to be waited at registers with this method. The second method is adaptive pipelining. This pipelining method is controlled from HDL Coder. There is a requirement that specifies the target platform for using adaptive pipelining. HDL Coder insert registers to block design if adaptive pipelining is enabled.

2.2. Hardware Design: Creating Custom IP with HDL Coder

The sound effect design is described in a Simulink model; whose used blocks must be supported by HDL Coder [12]. In this work, as a sound effect, a filter design is used additional to previous work. A distortion sound effect with no parameters was designed for previous work. In this work, this distortion effect is updated with pipelining registers and input parameters. The filter design, which is new sound effect, is made on FDA Filter Designer Tool, and this design converted to a Simulink block. The block that is generated uses basic elements such register, gain, addition and subtraction block, which is supported by HDL Coder.

When Simulink block design of sound effect is completed, custom IP block can be generated with that block using HDL Coder. Vivado 2020.2 provides the synthesis tool to HDL Coder for converting the block to IP core design in VHDL. The IP core runs on an FPGA inside of SoC design and uses AXI4-Lite interface in order to communicate with whole system.

2.3. Software Design: Hardware Management and User Interface

The designed custom IP core is controlled over a software application runs on an ARM processor. The application handles the basic processes. It makes the address mapping with BSP (Board Support Package) of the board that is used, routing the audio data between IP cores etc. In order to use custom IP core, driver of the custom IP has to be added to software and associated with application for accessing the IP core. The driver includes the address of required addresses.

Other duty of the software is to provide communication between system and user. The system and user communicate over UART (Universal asynchronous receiver-transmitter) at 115200 baud rates. The interface is managing from a basic console, but it can be improving for advance applications.

2.4. Design Environments

Sound effect is designed on Model Composer and System Generator 2020.2 provided by Xilinx, based on MATLAB & Simulink version 2020b. Model Composer provides the full compatibility for Xilinx boards. Some of the steps for introducing the board, synthesis settings are already defined with Model Composer. It gives setting up the system easier than normal MATLAB environment.

Created custom IP core is implemented to system on Vivado 2020.2. This Vivado version provides new software platform rather than previous work. This new software environment name is Vitis and is used 2020.2 version. Vitis comes up with some change compared to SDK, which was old software environment for Vivado. The application and platform are separated on Vitis, which makes the target platforms configurable.

III. METHODOLOGY AND IMPELEMENTATION

The main idea behind this work has already summarized in Section 2. In this section, all details and project implementation are covered. The path to the solution is described in subsections below.

3.1. Necessity of HW/SW Co-design Methodology

Software based DSP solutions for audio processing may be not enough to handle when desired processing time is real-time. For example, complex sound effects generate more latency while monitoring the input audio to an output. Hardware based solutions are solved this latency problem. But it is hard to implement and results in the loss of functionality advantage of software solutions. When HW/SW co-design methodology is used, the latency results will be so much better than software, and hardware still can be manipulated from the software. Therefore HW/SW co-design methodology offers a solution where is used advantages of both sides.

3.2. System-on-Chip Design Architecture with Sound Effects on ZedBoard

In this work, an SoC design is implemented on ZedBoard, which is provided by Xilinx. ZedBoard includes a Zynq-based programmable SoC platform. This platform works within an ARM processor core. The ZedBoard that we used contains ZC7020 chip.

SoC platform can be explained in two parts, PL (Programmable Logic) that was mentioned as hardware and PS (Processing System) that was mentioned as software. PL side is powered within an FPGA and PS side is run on an ARM processor. PL consists of PS, which can be shown in
Figure **2**. The IP cores are the changeable components of the PL which means that IP cores can be used in different designs again.

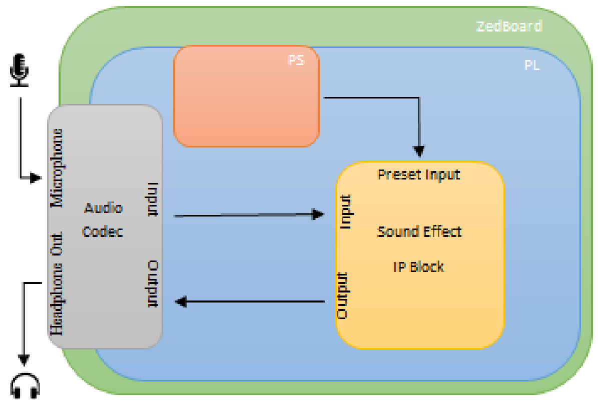


Figure 2. Overview of SoC Architecture

3.3. Sound Effect Design on MATLAB & Simulink

Two different sound effect designs are made on Simulink for this work. First design is related with previous work, which is distortion effect. Addition to previous design, parameter inputs for managing effect from software and pipelining for decreasing latency more were added. Seconds design is about a lowpass filter. Many more filter designs can be implemented like the way is going to explained below.

3.3.1. Distortion Effect Design

Distortion effect distorts the audio signal. It applies a gain to incoming audio signal and saturates that signal in order to keep signal in valid area with a saturation block [13]. In this version, there are 2 parameters for managing distortion effect. First parameter is drive, which provides the adjusting level of distortion level, and second parameter is mix, which provides how much percentage of clean and distorted audio is going to mixed. The design of distortion effect can be shown Figure 3. Components are described in Table 1.

Table 1. Distortion effect block explanations

Block	**Explanation**
audio_in	Audio signal incoming from MATLAB workspace
ConstantDrive	Input of drive parameter (0 minimum, 1 maximum), that sets intensity of the effect
ConstantMix	Input of mix parameter (0 minimum, 1 maximum), that sets rate of mixing with distorted and clean audio
distortion subsy	Distortion sound effect block
convert & double	Converting blocks between fixed point and double data types.
output.wav	Saves the audio to a multimedia file

As shown in Figure 3, distortion sound effect takes three input parameters and provides an output to obtain effected audio. Audio input is obtained from MATLAB workspace and output audio is saved to a wav file. The incoming data values and saved data values are in double data type. Distortion block performs the computation in fixed point data type with 16 bits length with 14-bit fraction length, which is proposed in [14]. When looking into the subsystem of distortion, there can be seen pipelining methodology is implemented. This methodology uses registers for accelerating the hardware. Delay blocks are used as register. They keep the corresponded data for one clock cycle of the system. It provides to store data in short distances between blocks where they have computation processes. When the pipelined design is implemented, there is no need to wait for completing the process for one sample, it can take more cycles in a one subsystem in sequential line. Implementing pipelined design increases the latency in theorical, but it improves the throughput, processing power and also system clock can be run at higher frequencies, which gives reducing latency values compared to the design without pipelining. Pipelined design of the distortion block is shown in Figure 4. There are several blocks and they are used mainly for multiplication, addition, saturation, delay and supplying a constant to another block. The delay blocks perform a wait operation for one clock cycle. These delay blocks must be added in right places in the design in order to making processes in parallel. Additional registers can be added for blocks with heavy processing load. z^{-1} block is a delay component and it represents a pipeline register with reconfigurable clock-cycle.

Registers are implemented as with these parameters: input pipeline is 1, output pipeline is 1. These parameters are provided from HDL Coder and, all used blocks are supported from HDL Coder toolbox in Simulink library. All multiplication, subtraction and addition processes are done sample based. In Figure 4, there can be imagined 5 register stages. Each register transmits the data on rising edge of a clock cycle of system to another register stage. Inside of the subsystem signals are used in signed fixed point data type with 16 bit data length and 14 bit fraction length suggested as in [14]. Besides, adaptive pipelining is also used in this block design.

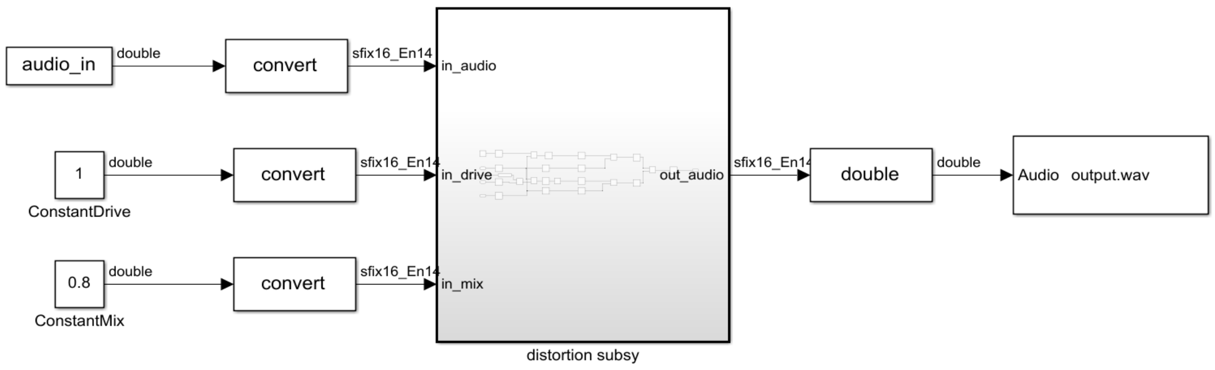


Figure 3. Simulink model of distortion effect

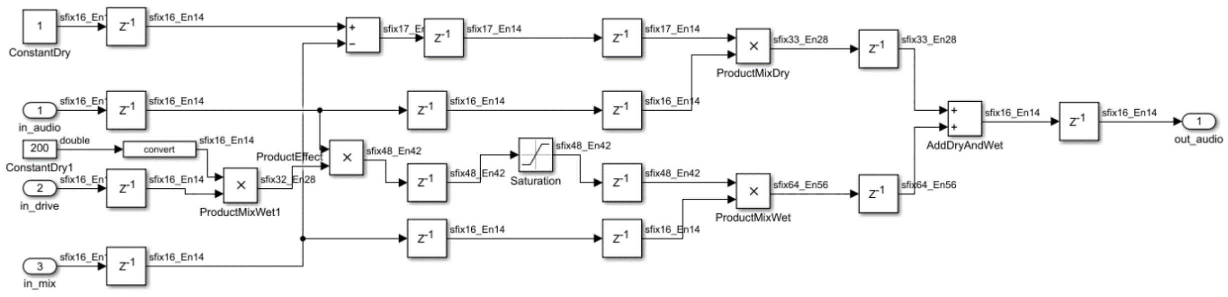
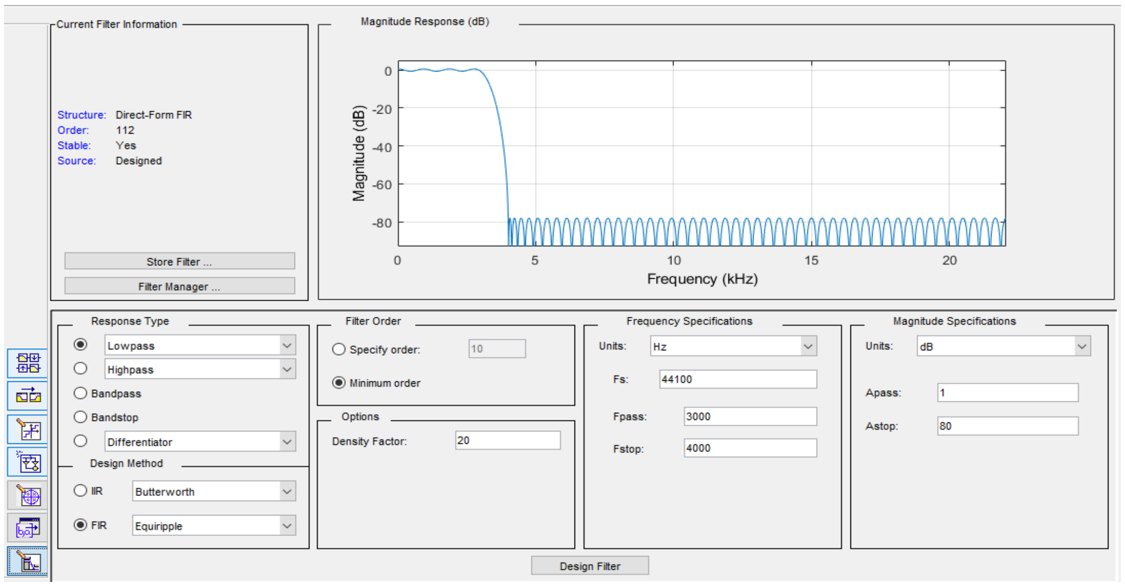


Figure 4. Subsystem of distortion effect in detail

3.3.2. Lowpass Filter Design

A lowpass filter design is created in order to show how Simulink gives ease of design for hardware implementations. This lowpass filter passes lower frequencies than 3 KHz and stops higher frequencies than 4 KHz where can be seen in Figure 5 as magnitude response graph. Filter Designer tool (as known as FDA Tool) is used for creating filter block. Filter designer can be used by entering filterDesigner to MATLAB command window. The tool will be opened after this process. The tool provides some parameters to select and it makes easier to design a filter. In this work, an equiripple FIR filter is used. Equripple filters are suited for specific tolerance for passing and stopping for some frequencies [15]. The order of design may be reduced with other design methods. It may reduce the latency more because reducing order means that less blocks are going to used. Since the methodology is emphasized here, the most efficient design has not been studied. The filter design in the Filter Designer is shown on Figure 5.

When filter design is finished, the design can be converted to a Simulink model from File > Export the Simulink Model selection. Input processing is selected as sample based and all optimization selections are checked before realized model. The block that is generated is added to another Simulink model. This block design is generated automatically, which is shown in Figure 7. In this design, there are 112 delay blocks which is corresponded to design order in the Filter Designer and they are used as register as well with 1 input and 1 output pipeline parameters. The gain blocks in the design enables to making element wise multiplication with a constant [16] defined from Filter Designer. The generated subsystem block is also shown as in Figure 6. The blocks that are shown in figure are described in Table 1 before, except filter_lowpass block. This block is used as lowpass filter sound effect and created from Filter Designer Tool as mentioned above.


Current Filter Information

Structure: Direct-Form FIR
Order: 112
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Butterworth [✓ FIR Equiripple]

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Density Factor: 20

Frequency Specifications: Units: Hz; Fs: 44100; Fpass: 3000; Fstop: 4000

Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

Figure 5. Filter design in the Filter Designer tool

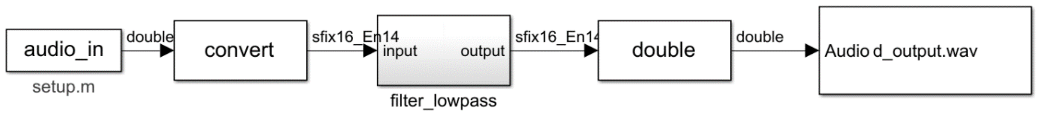


Figure 6. Filter block in the Simulink

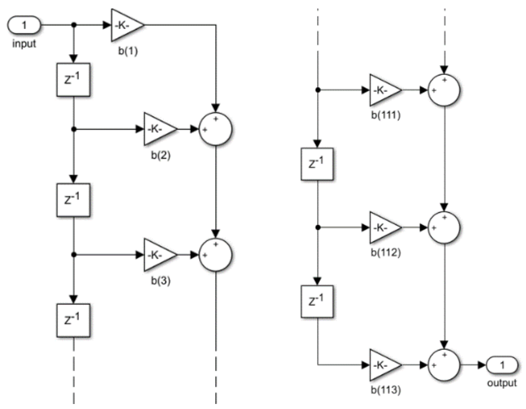


Figure 7. Inside of designed filter block in Simulink (dashed lines represent that blocks repeat in the same way from input to output)

The basic blocks in the design are supported from HDL Coder. Gain block parameter K has different values which are defined from Filter Designer tool. Lowpass filter block is selected as use adaptive pipelining from HDL Coder properties. Also, all delay blocks are used as register in order to implement pipelining for better execution times.

3.4. Creating Custom IP Cores via HDL Coder

After design of sound effect has completed, the sound effect block in the Simulink is converted to IP core via HDL Coder. Before converting the filter block to IP core, model and subsystem compatibility is checked with HDL Coder. These steps show if there is any error or warning for better design compatibility. Some of the warnings that is taken in that stage was about reset type of clock settings, changing block names etc. These checking tools gives facility of seeing any mistake before generating IP core.

HDL Workflow Advisor is a tool that is used for creating custom IP core provided by HDL Coder. This tool provides to selecting target platform, preparing the model for HDL code generation and generating IP core. Since Simulink is opened from Model Composer and System Generator, the board that is going to used is already set up to HDL Coder. It provides pass these processes faster than normal Simulink environment. The settings that are used for ZedBoard are described in Table 2. IP core is generated as used these parameters.

Table 2. HDL Workflow Adviser Parameters

Parameter	**Input**
Target workflow	IP Core Generation
Target platform	Generic Xilinx Plaform
Synthesis tool	Xilinx Vivado
Tool version	2020.2
Family	Zynq
Device	xc7z020
Package	clg484
Speed	-1
Processor/FPGA synchronization	Coprocessing - blocking
Target platform interfaces	AXI4-Lite
Language	VHDL
Adaptive pipelining	Checked

3.5. Adding Generated IP Core to a Hardware Design in Vivado

The hardware design is synthesized, implemented and routed using Vivado design suite (Vivado v2020.2). First, a block design is created, and this design uses ZYNQ7 Processing System IP core to instantiate ARM processor in the system. An audio codec IP core is used from Zynq Book Tutorials [11] to obtain audio signals from microphone inputs and giving back to the headphone output. All IP cores communicates with each other over AXI interconnect. Sound effect IP blocks must be added first to the repository. For this purpose, new IP repository is added where is located to generated IP core location in the IP Catalog, it adds IP's automatically from the repository. After adding process is done, IP's can be added to the block design. IP block connections are done automatically thanks to the Vivado. After synthesis, placing and route operations, a bitstream file is generated to program FPGA. The bitstream file must be exported to use this hardware design in composing software. Final block design of the PL part of the SoC architecture is shown in Figure 8.

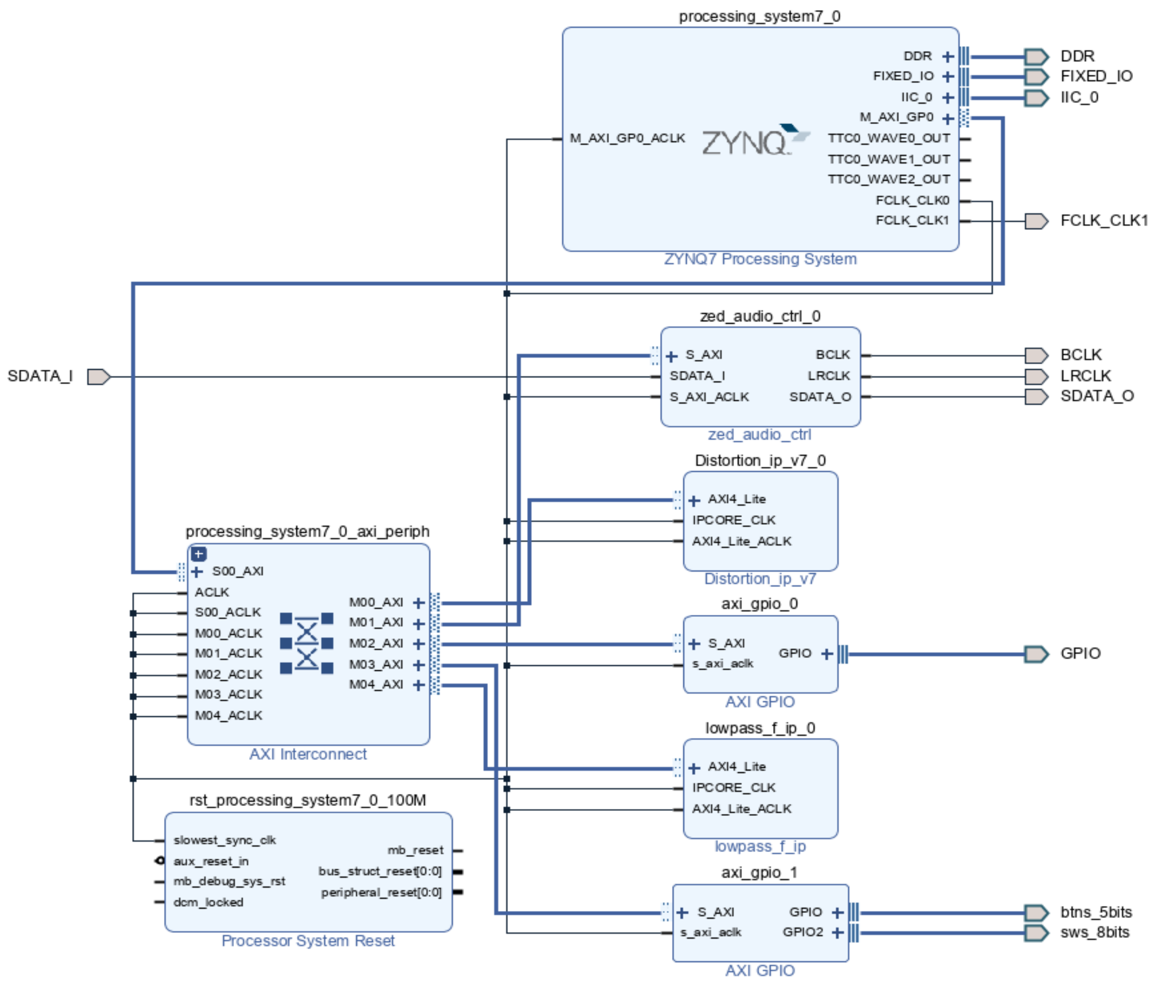


Figure 8. Block design of SoC architecture

3.6. Composing the Software on Vitis

Vitis, which is a recent and unified software development environment provided by Xilinx, is used for software part of the design. New application project and platform project are created first. Platform project is used to define the hardware specification as a platform. It defines hardware specifications such as drivers for hardware design etc. It is used bitstream of hardware, which is generated on Vivado, and it can be updated when hardware design is changed. On the other side, application project is the part of using hardware IP's and using algorithms in order to realize what is desired from software side. The application project must be created with this platform project instance. IP core drivers must be added to the application project for address definitions of corresponded IP cores.

The application project provides an UART connection in order to communicating with user. According to given input parameters to UART, application transmits the audio signal between the corresponding addresses of sound effect IP core and input & output registers of audio codec IP core. These addresses are determined in HDL Coder IP core generation step and defined in IP core drivers which are created automatically by HDL Coder. Also, input parameters are managed in here, which was mentioned before changing the behaviour of sound effect.

3.7. Internal Structure of Lowpass Filter Sound Effect IP Core

In the lowpass filter IP core there are 4 different modules. These modules are described in this subsection. Figure 9 illustrates the internal structure of lowpass filter IP core.

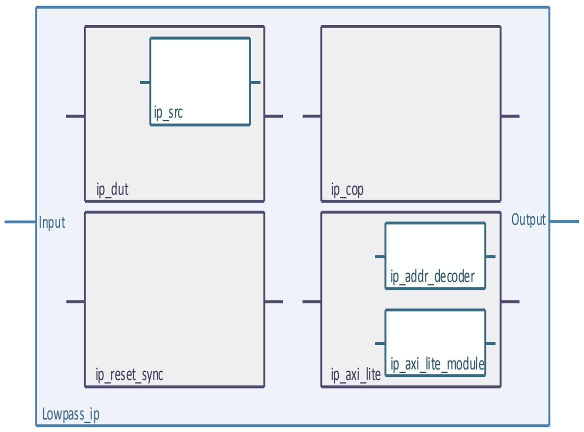


Figure 9. Illustration of Lowpass Filter IP Core in Detail

3.7.1. ip_dut

This submodule is used for processing the audio signal. It has one more component sub-module named ip_src which makes the processing operations and pipelining management. The input and output are fixed point data which was defined in Simulink design. In addition to these, reset, clock and enable signals goes as input to this block.

3.7.2. ip_reset_sync

This submodule manages the reset state of IP. When a reset signal is emitted to this IP core, it resets states for processing and pipelining processes.

3.7.3. ip_cop

This submodule is used as controller of processes. It manages the states of controller such as strobe, enable and ready signals. This module generates a set of signals to control (starting, enabling for write operation, etc.) the other hardware components in the design and also checks the ready signals to determine whether they finished the execution.

3.7.4. ip_axi_lite

This submodule manages the communication of hardware IP core with other IP's in the system through AXI4-Lite interface. It handles the necessary signals with cooperating other submodules that are mentioned above. It has two other submodules inside of itself as component. Input and output signals transfer from this submodule.

IV. EXPERIMENTAL RESULTS

4.1. Latency for Simulink Solutions

Everything described in the methodology section has been carried out on Simulink and ZedBoard. The latency results for Simulink models could not be obtained. Because the delay blocks that are used for pipelining are caused delaying in audio, which makes the simulation results meaningless. But the driver latency is already known from previous work. The latency result can be taken minimum value as 2.447 milliseconds without applying an effect and 2.8 milliseconds with distortion IP block from previous work with ASIO4ALL v2 driver, which can give an idea for comparison the results obtained from ZedBoard. These results were evaluated with a computer which has Intel i7 7700HQ processor, Realtek High Definition Audio driver version 6.0.1.8142, and audio codec ALC269 at 44.1 kHz.

4.2. Latency Calculation for ZedBoard

The latency values are calculated with a TTC timer provided by Zynq device on the ZedBoard. When a sample is processed, it is obtained the time value from physical timer counter register and stored previous value in different variable. The latency calculation is shown in Figure 10 and described below.

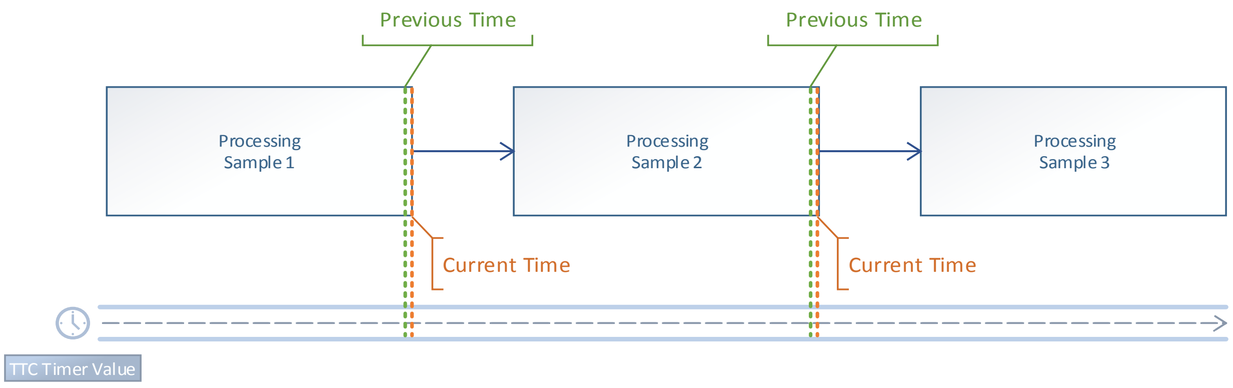


Figure 10. Calculating latency and representation of times

The algorithm of obtaining latency is described with a basic pseudo code below.

```
previous_time = current_time;

current_time = GetTime();

elapsed_time    =    current_time    –
previous_time;
```

If relation between the Figure 10 and code above is examined, previous time is the current time in the previous processed sample. Previous time is stored in another variable and elapsed time is calculated as difference of previous and current time. This method provides to obtain a latency value with time elapsed outside the IP core.

4.3. Latency Results for ZedBoard Implementation

Different implementations are tested on ZedBoard. Distortion sound effect and lowpass filter sound effect are implemented together and one by one on hardware. The test specifications and results are described in Table 3. The average values are obtained from 10 test records.

Table 3. Test specifications and results implemented on ZedBoard

Sound Effect(s)	Latency from Pipelining (cycle)	Clock Frequency	Average Clock Cycle Spent	Average Latency (nano-seconds)
Distortion + Lowpass Filter	-	100 MHz	710	1065
Distortion	12	150 MHz	588	882
Lowpass Filter	226	100 MHz	714	1071

Clock frequency shows the frequency that can be reached at maximum value for the designed sound effect IP block. Results show while increasing clock frequency, latency is decreasing. The way of increasing the clock frequency is using pipelined register designs. But if these pipelined registers are increased more than enough, that would be cost of more latency and this puts the results at a disadvantage instead of advantage. Different sound effects have almost same latency values at the same clock frequency.

Unlike similar studies, in this study specific latency values are obtained and compared by using MATLAB & Simulink, which makes much easier to design development and saves time, for IP core design with pipelining technique, including previous study [4]. Significant improvement in latency has been demonstrated. Using pipelining on complex Simulink designs make it synthesizable to an IP Core that is run at desired clock frequencies. This gives the opportunity of processing an audio in a specific time.

As a result, designs with pipelined registers reduce the latency. The best latency values are under the 1 microsecond (882 nanoseconds). There is 12.75% decreasing in latency value for distortion block with pipelined design compared to previous work. Much better Simulink designs can be made that it is opened to development for less latency values. This study also revealed how easy it is and showed have good performance values for designing DSP system by using MATLAB & Simulink.

V. CONCLUSION

This study shows the advantages of HW/SW co-design methodology with a system-on-chip design on audio processing. Very small latency values can be obtained with using specialized hardware while controlling it over a software application for DSP systems without losing any functionality of software. Development time and design productivity significantly improved by making block-based designs in MATLAB & Simulink. High-level DSP systems can be implemented easily with this methodology. For sound-effect applications, the followed methodology can be a viable solution on real time applications. In this context, neural network-based audio synthesis algorithms can also be accelerated as a future work to provide real-time performance via similar design methodology with MATLAB or other high-level synthesis tools.

REFERENCES

[1] Blackledge, J. (2006). In Digital Signal Processing (2nd ed.). Chichester: Horwood Publishing.

[2] Pirkle, W. C. (2019). Designing audio effect plugins in C++: for AAX, AU, and VST3 with DSP theory (2nd ed.). New York, NY, USA: Routledge.

[3] Elhossini, A., Areibi, S., and Dony, R. (2006). An FPGA Implementation of the LMS Adaptive Filter for Audio Processing. *2006 IEEE International Conference on Reconfigurable Computing and FPGA's (ReConFig 2006).* San Luis Potosi, Mexico: IEEE.

[4] Esen, Y. E., and San, İ. (2020). LowLAG: Low-latency hardware accelerator of a sound effect with system-on-chip design. *ASYU 2020, Innovations in Intelligent Systems and Applications Conference.* İstanbul.

[5] Juillerat, N., Arisona S. M., and Schubiger-Banz, S. (2007). REAL-TIME, LOW LATENCY AUDIO PROCESSING IN JAVA. *Proceeding of the International Computer Music Conference.* Copenhagen.

[6] Meyer-Baese, U. (2007). Digital Signal Processing with Field Programmable Gate Arrays (2nd ed.). Berlin: Springer.

[7] Audio quailty on networked systems, Yamaha, https://uk.yamaha.com/en/products/contents/proaudio/docs/audio_quality/05_audio_quality.html (June 2020)

[8] Pfaff, M., Malzner, D., Seifert, J., Traxler, J., Weber, H., and Wiendl, G. (2007). IMPLEMENTING DIGITAL AUDIO EFFECTS USING A HARDWARE/SOFTWARE. *10th Int. Conference on Digital Audio Effects (DAFx-07).* Bordeaux.

[9] Byun, K., Kwon, Y., Park, S., and Eum N. (2009). Digital Audio Effect System-on-a-Chip Based on Embedded DSP Core. *ETRI Journal.*

[10] Byun, K., Kwon Y., Koo B., Eum N., Jeong K. and Koo J.. (2009). Implementation of digital audio effect SoC. *2009 IEEE International Conference on Multimedia and Expo.* New York.

[11] Crockett, L. H., Elliot, R. A., Enderwitz, M. A., and Stewart, R. W. (2014). The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC (First ed.). Glasgow: Strathclyde Academic Media.

[12] HDL Coder™ Getting Started Guide, The MathWorks, Inc., https://www.mathworks.com/help/pdf_doc/hdlcoder/hdlcoder_gs.pdf (June 2020)

[13] Limit input signal to the upper and lower saturation values, The MathWorks, Inc., https://www.mathworks.com/help/releases/R2018b/simulink/slref/saturation.html (June 2020)

[14] Fixed-Point Conversion. In HDL Coder™ User's Guide, The MathWorks, Inc., https://ww2.mathworks.cn/help/pdf_doc/hdlcoder/hdlcoder_ug.pdf (June 2020)

[15] Signal Processing Toolbox™ User's Guide, The MathWorks Inc., https://www.mathworks.com/help/pdf_doc/signal/signal.pdf (December 2021)

[16] Multiply input by constant. The MathWorks Inc., https://www.mathworks.com/help/releases/R2018b/simulink/slref/gain.html (June 2020)