Aims and Objectives

In this project, I addressed it from two perspectives: academic aims and technological objectives. During the academic portion, I gained knowledge and design expertise with reconfigurable hardware platforms for Field-Programmable Gate Arrays (FPGAs). The most intriguing aspect is that Hardware Description Language (HDL) delivers superior performance, flexibility, parallelism or concurrency, and energy efficiency in digital system designs when compared to other popular platforms such as Java and C. Second, as part of the technical objective, I designed and implemented a finite impulse response (FIR) filter utilising Terasic's Cyclone V (5CSEMA5F31C6N) DE1-SoC development board. The entire system is coded in Very High-Speed Integrated Circuits. Hardware Description Language (VHDL) and IEEE standard 1076-1933 libraries.

Objectively, I initially concentrated on the three major sub-blocks: codec initialization, serial-to-parallel (S2P) adaptor, and finite impulse response (FIR) block. Each block is coded in Quartus Prime Lite Edition 23.1 and generates VHD and VHT files, which may then be tested and analysed using Questa-Intel FPGA Starter Edition 2023.3 or ModelSim HDL Simulator. Second, I integrated all of these blocks, assigned a PIN configuration using the Pin Planer tool, and programmed the FPGA chip. Ensure and validate that each block design works as planned. Test various real-time functions and interfaces to confirm that everything works properly, plot the FIR filter's frequency characteristics graph, and comprehend their cut-off frequency range.

Introduction

Design a finite impulse response (FIR) filter using FPGA technology. There is a chance that FPGA algorithms will increase speed and decrease latency. The FPGA devices are specifically designed for total device power optimisation and make use of the ARM Cortex-A9 dual-core family of Cyclone V (5CSEMA5F31C6N) devices, which are well-known for their remarkable performance and density. Furthermore, Altera's unique redundancy technology lowers component costs and increases throughput dramatically, guaranteeing better performance and signal integrity [1]. An ideal filter is a network that permits signals of a specific frequency to pass while blocking others. They are classified as low-pass, high-pass, band-pass, band-reject, and all-pass according to the range of frequencies that are permitted through or not. In present-day computing applications, the term filter is a crucial part of the electronic system and is commonly used from the basic products of radio and television sets to the latest digital signal processing (DSP) components [2].

The FIR filter is now widely used in the realm of digital signal processing (DSP), instrumentation, and telecommunications. Finite Impulse Response (FIR) filters have widespread application. Because of the high-performance requirements of large applications, the system's complexity has gradually increased. As a result, many taps are required for processing high sampling rates, and moreover, operations are computationally intensive and more complex than with traditional filters. Nowadays, FIR filters make a remarkable difference on the market in terms of performance, stability, flexibility, and reliability. At the same time, the cost of production is reasonable, and energy consumption is reduced [3]. The block diagram of the FIR filter is shown in Figure 1 [4], and the graphical representation is shown in Figure 2 [5].

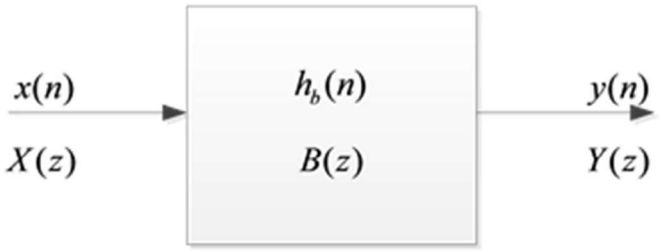

Figure 1. Block diagram of Finite impulse Response (FIR) filter [4]

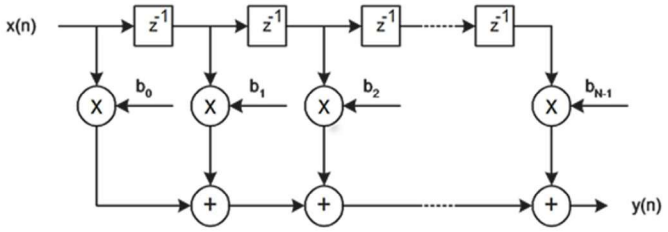

Figure 2. Graphical representation of Finite impulse Response (FIR) filter [5]

During the past 20 years, there has been noticeable progress in FPGA cost reduction, as measured by the International Technology Roadmap for Semiconductors (ITRS). This system's functionality is centred on raising the FPGA devices' price and efficiency. Ensuring these factors foster innovation in a variety of technology domains, including developing research materials, microelectromechanical systems, process integration, devices, and structures, as well as system drivers and design [6], FPGAs that use dedicated hardware resources can efficiently achieve application-specific integrated circuit (ASIC)-like performance while lowering development time, cost, and risk. This is especially true when it comes to the FIR filter implementation in FPGA, which plays a crucial part in ITRS. [7] [8]. Because of its increased versatility, the FPGA has consequently emerged as the top option for signal processing system design.

In terms of implementation progress and group management, we divided our work into different parts, mainly codec, serial to parallel (S2P), and FIR filter. Initially, we aim to develop the VHDL programme code as per the provided template for codec and S2P. However, the FIR part is implemented based on the provided design specification, such as a 16-bit, 41 kHz analogue input/output audio stream, eight taps, and one hardware multiplayer. After completion of each part, we are ensuring the output waveform is correct as we expected with certain stimuli in the test bench condition. Finally, we combine all three blocks and programmes on FPGA chips and verify their performance characteristics.

Methodology

It outlines the framework within which research is conducted. I used the below steps to achieve the result.

1. Collecting technical specification data for the codec chip (WM8731/WM8731L) and Cyclone v (5CSEMA5F31C6N) FPGA chip.
2. Implementation method and understanding their interfacing and control.
3. Signal refinement and validation.
4. List of hardware and software components.

In step 1, the technical features of the WM8731/WM8731L codec provide high-quality stereo audio performance and are appropriate for high-end computing disciplines. They ensure perfect audio reproduction by using a 24-bit ADC and DAC for sampling rates ranging from 4 kHz to 96 kHz. Both the input and output interfaces are adaptable and offer a variety of connectivity features, including I2S (inter-IC sound) and DSP (digital signal processor) modes. I2S is a standard serial bus protocol that transmits digital audio data between integrated circuits (ICs). It is composed of three lines: serial clock (SCK), world clock (WS), and serial data (SD). The codec's distinctive feature provides accurate audio using signal-to-noise ratio (SNR) and total harmonic distortion plus noise (THD+N). The operative voltage is a single 3.3V supply and control via the Inter-Integrated Circuit (I2C) interface for easy configuration [9]. In the context of FPGA Cyclone v (5CSEMA5F31C6N), we offer efficient implementation of custom digital circuits and high-performance signal processing algorithms. It provides a range of input/output interfaces and various configuration methods. Designed for power efficiency and built-in security features, it is well-suited for embedded applications, signal processing tasks, and prototyping projects.

In step 2, the implementation phase, I divided the whole project into three different subparts, such as the codec (analogueue-to-digital converter and digital-to-analogueue converter), the S2P (Serial to Parallel and Parallel to Serial) converter, and the FIR (Finite Impulse Response) filter. Each part was separately coded in the VHDL programming language and tested with Quartus Prime software, which really helped to reduce the complexity of the entire project. After that, understand the work concept between the codec and the FPGA. The main processing unit is responsible for digital signal processing tasks and executing the FIR filter algorithm. The codec chip acts as an interface between analog-to-digidigital (ADC) and digitalanalog convers conversion. The communication between the FPGA and codec occurs via standard protocols like I2C for the control interface and S2P for the serial audio interface. Typically, the FPGA and the codec chip are the primary hardware components involved in the signal processing system. As shown in figure 3 [10], the block diagram of the De1-SoC board.

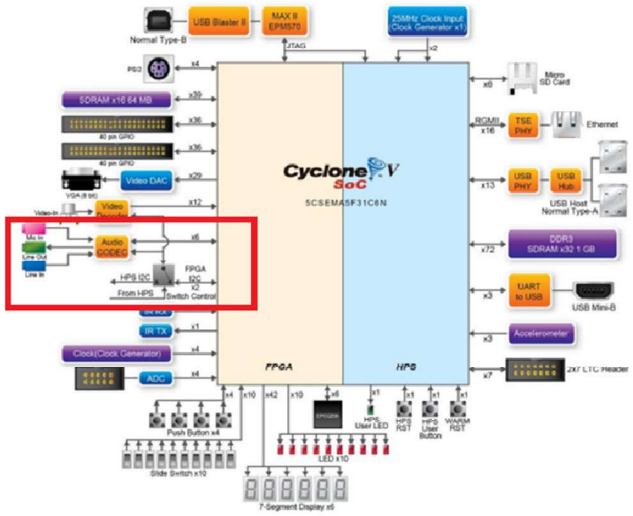

Figure 3. Block diagram of DE1-SoC board (Codec and FPGA chips) [10]

In step 3, the signal refinement and validation serve as testing of each subpart, such as the codec, S2P, and FIR. For the verification part of the analysis, I created the test bench file in each subpart and stimulated the signal conditions, such as 50 MHz clock generation, reset, and other essential signals as well. Finally, ensure the all-signal port map has the correct signals. During testing, it is confirmed that all signals are correctly generated, and data is transferred using I2C interfacing. After testing and verification of all subparts, I compiled the individual code into a single file, loaded it onto a DE1-SoC board, and examined waveforms with Signal Tap. First, I configured the pins and built the design using FPGA software. Then, programme the FPGA, making sure that Signal Tap is enabled to monitor signals. Run the design on the board and capture waveforms with Signal Tap software, then examine the data to ensure everything works well. This technique allows us to completely test and verify the FPGA design on the DE1-SoC board. In step 4, please see below Table 1 for the required hardware and software components in the project.

Sl.No	**Hardware**
1	DE1-SoC development board
	USB Cable (Type A to B) for FPGA programming and control
	USB Cable (Type A to Mini-B) for UART control
	12V DC power adapter
2	Waveform Generator(Model no: 33521A Agilent-30MHz)
3	Digital Storage Oscilloscope (Model no: DSO5014A Agilent)
4	Connecting probes
5	Computer/Laptop
	Software
1	Quartus Prime Lite Edition 23.1
2	Questa – Intel FPGA Starter Edition 2023.3 or ModelSim HDL Simulator

Table 1. List of Hardware and Software components used.

Design Specification and refinement

The design specification and refinement of this project implement an 8-tap FIR filter on an FPGA (Cyclone V on a Terasic DE1-SoC board) to process analogue audio signals from a WM8731 codec, as condensed by certain steps below.

1. Setup and Initialization
2. FIR filter design
3. FPGA implementation
4. Verification and testing
5. Optimization and refinement

The configuration setup of the codec (WM8731/WM8731L) for audio processing ensures good compatibility, I2C communication, and input/output analogue gnals. However, it provides high-quality audio computing capabilities and supports the configuration features of line mode, gain setting, and sampling formats. The I2C communication is established between the codec chip (WM8731/WM8731L), FPGA (Cyclone V-5CSEMA5F31C6N), and I2C controller to transmit control data to the codec's registers. These registers control various operations such as input/output modes, gain, and audio format. Finally, data read/write configuration is sent to the codec registers via the I2C bus from the FPGA. It controls the necessary bits to enable line mode, set gain, and select CD sampling format. Please see table 2 below for codec register map configuration.

REGISTER	BIT[8]	BIT[7]	BIT[6]	BIT[5]	BIT[4]	BIT[3]	BIT[2]	BIT[1]	BIT[0]	DEFAULT
R0 (00h) Left Line In	LRINBOTH	LINMUTE	0	0	LINVOL[4:0]					0_1001_0111
R1 (01h) Right Line In	RLINBOTH	RINMUTE	0	0	RINVOL[4:0]					0_1001_0111
R2 (02h) Left Headphone Out	LRHPBOTH	LZCEN	LHPVOL[6:0]							0_0111_1001
R1 (01h) Right Headphone Out	RLHPBOTH	RZCEN	RHPVOL[6:0]							0_0111_1001
R4 (04h) Analogue Audio Path Control	0	SIDEATT[1:0]		SIDETONE	DACSEL	BYPASS	INSEL	MUTEMIC	MICBOOST	0_0000_1010
R5 (05h) Digital Audio Path Control	0	0	0	0	HPOR	DACMU	DEEMPH[1:0]		ADCHPD	0_0000_1000
R6 (06h) Power Down Control	0	POWEROFF	CLKOUTPD	OSCPD	OUTPD	DACPD	ADCPD	MICPD	LINEINPD	0_1001_1111
R7 (07h) Digital Audio Interface Format	0	BCLKINV	MS	LRSWAP	LRP	IWL[1:0]		FORMAT[1:0]		0_1001_1111
R8 (08h) Sampling Control	0	CLKODIV2	CLKIDIV2	SR[3:0]				BOSR	USB/ NORMAL	0_0000_0000
R9 (09h) Active Control	0	0	0	0	0	0	0	0	Active	0_0000_0000
R15 (0Fh) Reset	RESET[8:0]									not reset

Table 2. Codec register map configuration.

Setting up the clock parameters ensures proper synchronisation and accurate processing of the audio data, like a "16-bit, 44.1 kHz digital stream." This statement is meant by the fact that the audio is sampled at a rate of 44.1 kHz, and each sample is represented using 16 bits. However, we need to ensure all codec internal operations are properly synchronised with the sampling rate of digital audio data. In the codec chip (WM8731/WM8731L), the oversampling rate is 256 in normal conditions, so for every output audio sample, the codec processes 256 input audio samples internally. Therefore, it correctly aligns with the sampling frequency, setting the MCLK frequency to 11.2896 MHz to achieve a 44.1 kHz sampling rate when configuring the WM8731 codec for audio processing (MCLK = sampling rate * 256 = 44.1 kHz * 256 = 11.2896 MHz). Please see below Figure 4 of the of the CODEC functional block diagram.

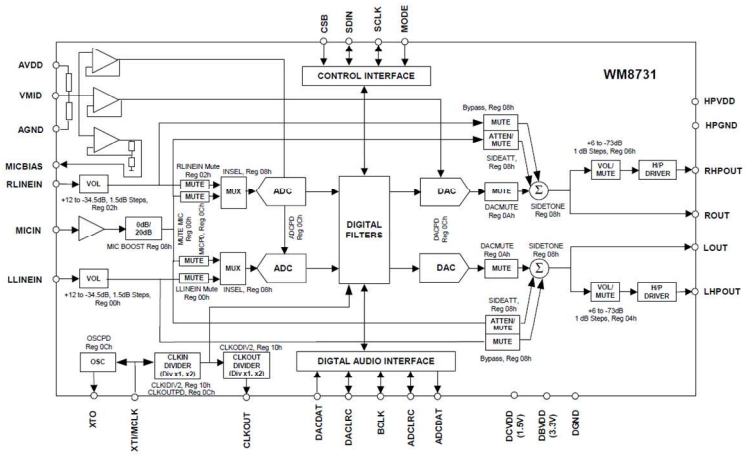

Figure 4 CODEC functional block diagram [9]

For the filtering operation of the FIR filter, the provided numerical coefficients are: 1260, 7827, 12471, 16384, 12471, 7827, and -1260. These coefficients determine the filter's behaviour and are applied to the input samples during the filtering process. However, by implementing one 16-bit multiplier, it aligns with each input sample by the corresponding filter coefficient. Finally, a 7-stage data shifter is employed, and each stage of the data shifter consists of a 16-bit register. The 16-bit multiplier's multiplied results must be shifted and accumulated over eight FIR filter taps, and this is done by the data shifter.

For FPGA implementation, code the VHDL language in the codec, S2P, and FIR filter parts and convert them into the HDL. Ensuring a 16-bit parallel interface between the codec and the FPGA chip's, and finally, clock synchronization. In the case of verification and testing, generate each VHT file (CODEC, S2P, and FIR) and stimulate the conditions. Perform the real-time experiment, ensuring the audio signal is correctly converted into the 16-bit, 44.1 kHz digital streams. Finally, measure the frequency response using the CRO and analyse the filter's performance. Moreover, in the optimisation part, carefully note the processing speed and refine the value of filter coefficients. Please see table 3 below for technical data specifications.

1	**Codec Configuration**: Analogue I/O Line mode, gain 1 unit, CD sampling format for left channel.
2	**Interfaces:** I2C for codec control, I2S for digital audio interface, 16-bit parallel interface for input and output.
3	**FIR Filter:** 8-tap filter with provided coefficients, implemented with one 16-bit multiplier and a 7-stage data shifter.
4	**Clocking**: MCLK/AUD_XCK configured as per codec datasheet
5	**Sampling Rate:** 16-bit, 44.1kHz digital audio streams.

Table 3. Technical data specification

Block 1: Serial to parallel

Input channel

In signal functionality, processes are triggered by the rising edge of CLOCK_50 (50 MHz) and the active condition of RST_N. In this moment, all internal signals and counters are initialised. First, the rising edge of the AUD_BCLK signal indicates the start of a new bit transmission in a serial audio stream. The serial audio data is received as AUD_ADCDAT with a clock signal

of AUD_ADCLRCK. The received serial data is extracted into a parallel data stream based on the rising edge of AUD_BCLK. These data were stored in registers and initialised the counter; the first bit of the AUD_ADCDAT is MSB (Most Significant Bit). If the counter value is greater than or equal to zero, the data transmission is ongoing (serial to parallel). This counter is decremented to keep track of the remaining data bits. When the counter value reaches zero, it means all data has been received, and after that, the ADCstb (ADC-strobe) signal is asserted for external synchronization. This synchronisation indicates the parallel data in ADCDAT is valid, and the later ADCstb signal is de-asserted with a small delay. The ADCrdy signal is an active handshaking mechanism between the adaptor and FIR filter block. See Figure 6 for the serial-to-parallel conversion (input channel) of the S2P adaptor simulation waveform. See figure 5 for serial to parallel block representation in the S2P adaptor and table 4 for the serial to parallel pin list.

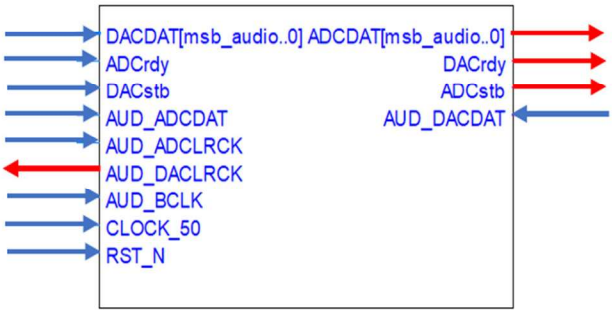

Figure 5. Serial to Parallel block representation in S2P adaptor.

1	AUD_ADCDAT: Serial data input from the ADC
2	AUD_ADCLRCK: Strobe signal indicating the validity of serial data from the ADC
3	AUD_BCLK: Clock signal for the serial interface
4	RST_N: Reset signal
5	ADCstrb- ADC strobe signal

Table 4. Serial to Parallel input channel pin list

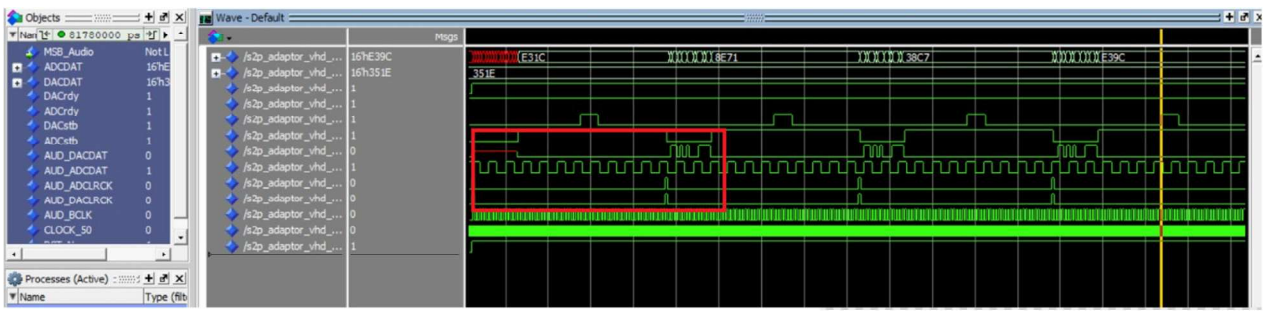

Figure 6. Serial to parallel conversion (input channel) of S2P adaptor simulation waveform.

To transfer digital audio data across different audio equipment, including sound cards, digital mixers, audio processors, and audio interfaces, a digital audio interface is a communication protocol or standard. Digital audio interfaces translate analogue audio impulses into digital data and back again while preserving the integrity of the audio stream. Figure 7. Digital audio interface [9].

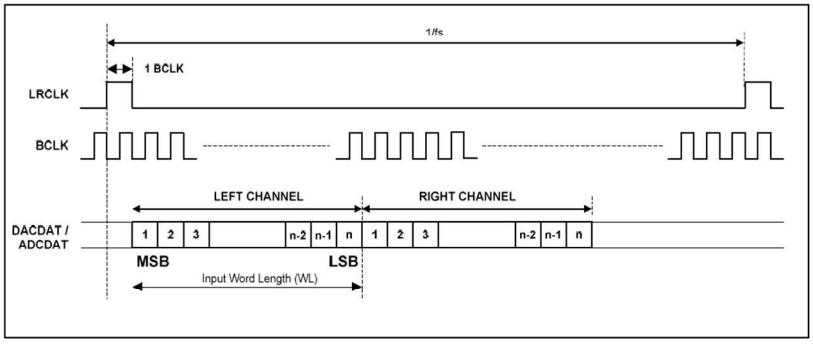

Figure 7. Digital audio interface [9].

Block 2: Finite Impulse Response (FIR) filter.

The Finite Impulse Response (FIR) filter is the fundamental part of this project and is used for filtering signals to achieve desired characteristics. In this real-time FIR filter, the incoming ADC data is processed and converted to the filtered DAC output. This function is employed by the shift register for storing past samples, and finally, the accumulator holds the summing products of samples and coefficients with appropriate control logic. The FIR filter is structured sequentially to compute the filtered output and coefficients. By convolving the input signal with the filter coefficients, the FIR filter achieves the desired filtering effect, making it suitable for various signal processing applications. See figure 8 for the FIR filter block with control signals.

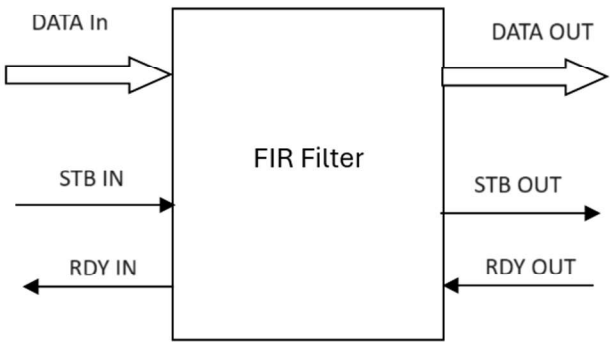

Figure 8. FIR filter block with control signals

The FIR filter equation is the output ‘y(n)’ at time ‘n’ with input signal x(n) and filter coefficient b(k), where k ranges from 0 to M. See mathematical formula 1 for the FIR filter equation below.

$$y(n) = \sum_{k=0}^{M} x(n-k)b(k)$$

y(n)= -1260 x(n) + 7827 x(n- 1) + 12471 x(n- 2) + 16384 x(n- 3) + 16384 x(n- 4) + 12471zx(n- 5) + 7827 x(n- 6) + -1260 x(n- 7).

Mathematical formula 1. FIR filter equation

For better filtering output, it always ensures some parameter, such as when adding two-word length data, the size increases by one bit, providing good precision. Preventing overflow accumulation needs 35 bits and output 16-bit compatibility. Finally, signal handshaking ensures data integrity, and time-multiplexing optimises the resources. Control Signals: STBin, RDYin, RDYout, and STBout coordinate data flow; signals synchronise operations and maintain stability. Once input data arrives, the processing phase begins, involving input shifting and accumulator resetting. Subsequently, multiplication and accumulation operations are executed. Finally, the resultant data is transferred to the output for further transmission. However, the FIR filter efficiently handles data, controls operations, communicates with interfaces, and ensures the ensures the stability of signal processing. See figures below for the for the data path of the FIR filter design (Figure 9) and the frequency response of the FIR filter (Figure 10).

1	AUD_ADCDAT: Serial data input from the ADC (MSB_Audio downto 0)
2	DACDAT: Parallel data input for the DAC (MSB_Audio downto 0)
3	ADCrdy: Signal indicating ADC readiness
4	DACrdy: Signal indicating DAC readiness
5	ADCstb: ADC strobe
6	DACstb: DAC strobe

Table 5. FIR Filter pin configuration

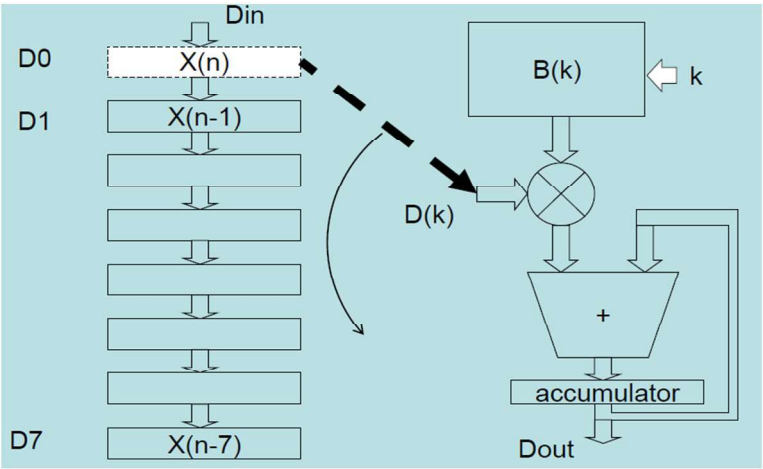

Figure 9. Data path of FIR filter design

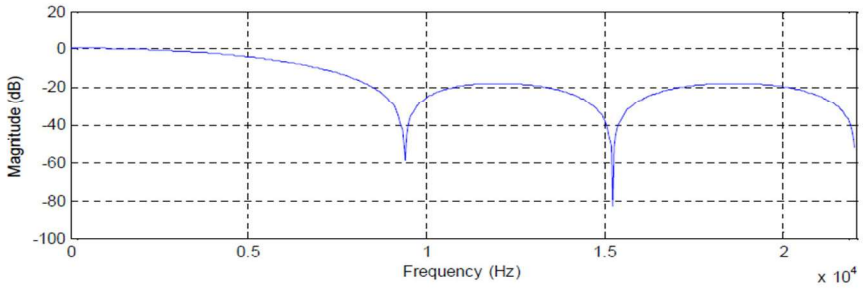

Figure 10. Frequency response of the FIR filter

The signals ADCrdy and DACrdy are used in the handshaking mechanism between the FIR filter and the S2P adaptor. If ADCrdy is high, the filter indicates it is ready to receive new digital audio data on the ADCDAT input port. This signal continuously asserted itself, except during the reset condition. The signal DACrdy indicates that previous data processing is finished and ready to accept new data from ADCDAT. However, the FIR filter continuously tells the S2P adaptor that it is ready to receive new data (indicated by the ADCrdy signal). When new data arrives, the ADCstb is indicated and stored temporarily in registers. The filter performs the operation, such as multiplying pre-defined coefficients with corresponding data points from the shift register and accumulating the product (removing the unwanted noise). Finally, the most significant bits (16 bits) of the accumulated value are passed to the output DACDAT. Once the data is received in DACDAT, the signal DACrdy indicates that previous data processing is finished and ready for new data from ADCDAT. See table 5 for signal pin configuration and figure 11 for the FIR filter simulation waveform.

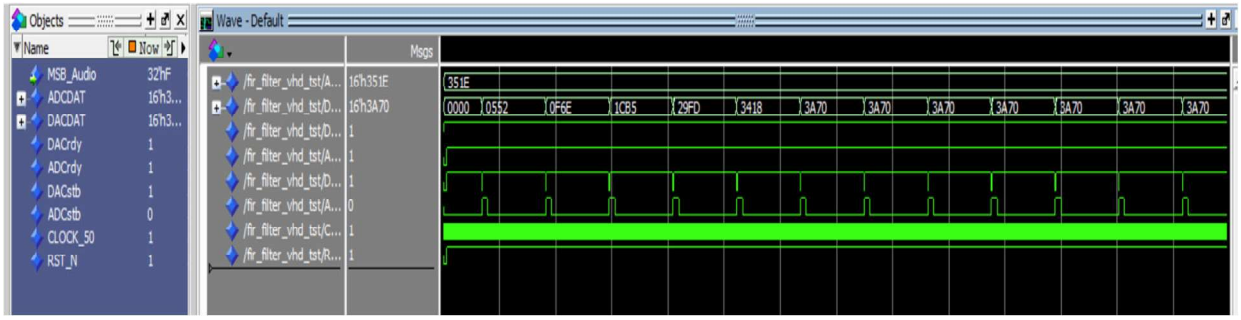

Figure 11. FIR filter simulation waveform

Block 3: Parallel to Serial converter

Output channel

The parallel-to-serial converter is employed to convert the audio data from parallel format within the FPGA into a serial data stream and is compactable in the codec chip. However, the overall functionality is based on some signal conditions. See figure 12 for parallel to serial converter and table 6 for port configuration.

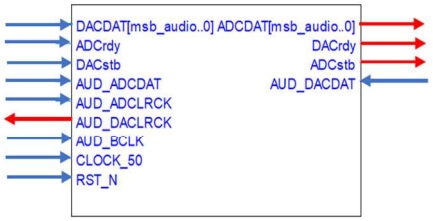

Figure 12. Parallel to serial convertor in S2P adaptor.

1	AUD_ADCDAT: Serial data input from the ADC
2	AUD_ADCLRCK: Strobe signal indicating the validity of serial data from the ADC
3	AUD_BCLK: Clock signal for the serial interface
4	RST_N: Reset signal
5	ADCstrb- ADC strobe signal

Table 6. Port configuration

Output channel.

In the output channel, the same parallel protocol was used for implementing parallel-to-serial conversion. First, the falling edge of the AUD_BCLK signal indicates the start of a new bit transmission in a parallel audio stream. The parallel audio data is received by DACDAT with a clock signal of AUD_DACLRCK. The received parallel data is extracted into a serial data stream based on the falling edge of AUD_BCLK. These data were stored in registers and initialised the counter; the first bit of the AUD_DACDAT is MSB (Most Significant Bit). If the counter value is greater than or equal to zero, the data transmission is ongoing (parallel to serial). This counter is decremented to keep track of the remaining data bits. When the counter value reaches zero, it means all data has been received, and after that, the DACstb (DAC-strobe) signal is asserted for external synchronization. This synchronisation indicates the serial data in DACDAT is valid, and the later DACstb signal is de-asserted with a small delay. Finally, the DACrdy signal is an active handshaking mechanism between the adaptor and FIR filter block. See figure 13 for parallel-to-serial conversion (output channel) in the S2P adaptor simulation waveform.

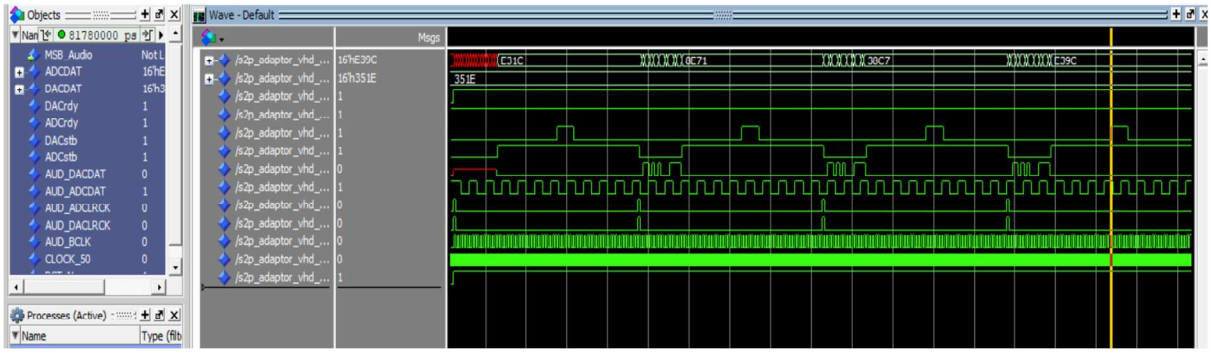

Figure 13. The parallel-to-serial conversion (output channel) in the S2P adaptor simulation waveform.

System of the block

The system of blocks explains the overall working procedures of codec, S2P, and FIR filter systems. See figure 14 below for system architecture. The codec is the starting block for signal receiving and outgoing, and this communication is based on the serial and control interfaces. See figure 15 for the for the codec simulation waveform. Secondly, the S2P adaptor receives the serial digital data stream from the codec and converts it to parallel for more efficient processing (Figure 6). This parallel data is stored in registers and fed to the FIR filter. Once the FIR filter operation, such as coefficient product multiplication and accumulation, is done and stored in registers (Figure 11), Again, S2P fetches this processed data from the appropriate registers and converts it to a serial data stream that is compactable for the codec chip (Figure 13). The communication between the S2P and FIR filters is accomplished due to the handshaking principle. Once the data is ready for processing, the S2P adaptor asserts the ADC ready (ADCrdy) signal to inform the FIR filter. Finally, after completion of processing the FIR filter, assert the DAC-ready (DACrdy) signal to S2P. It means the filter is ready to receive the next data stream for processing. These types of communication synchronisation avoid data loss and overfitting on both sides.

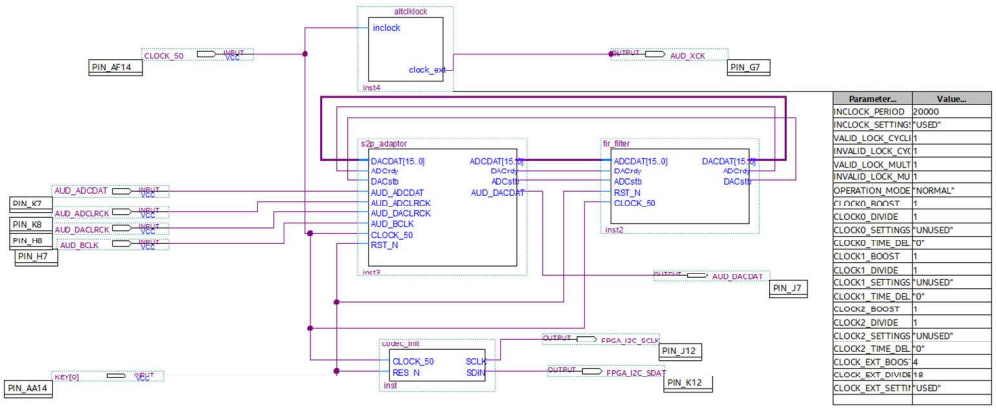

Figure 14. System Architecture

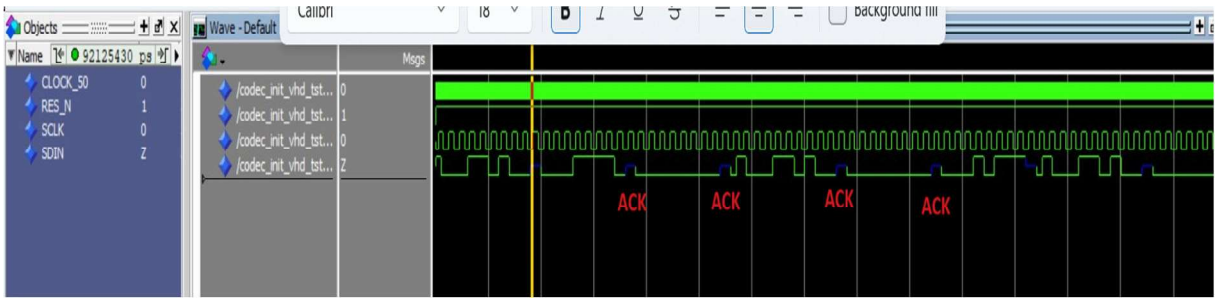

Figure 15. Codec simulation waveform.

Hardware experiment and Debugging

The physical experiment I carried out in the two parts. Part one is individually developing the VHDL code for the Codec, S2P, and FIR filters and verifying each signal condition state as I expected with the support of a test bench file from the Questa-intel FPGA starter kit. After testing and verification of all these subblocks, I step to part 2. In Part 2, I did the FPGA implementation using the DE1-SoC board. Firstly, I add all these VHD files (codec, S2P, and FIR) to a new project named Audio_Filter. Secondly, create a block diagram or schematic file, add these blocks, and assign all connections with pin assignments. See figure 11 for system architecture. (For more reference, see figure A: Configuration for pin assignment and figure B: Top View-Wire Bond Cyclone V-5CSEMA5F316 are attached in the Appendix Part. After device programming on the DE1-SoC board, connect the input signal from the function generator to the line-in port of the DE1-SoC board, and similarly, connect the connect the output line-out port to the oscilloscope for waveform analysis. The key [0] initialises the reset button on the board. Finally, measure (frequency and amplitude) the filtered waveform for a range of frequencies from 1 kHz to 25 kHz and plot the frequency response curve. See the following figures 16, 17, 18, 19, and 20 for the frequency response of the FIR filter waveform for inputs of 5 kHz, 10 kHz, 20 kHz, 21 kHz, and 25 kHz.

Figure 16. FIR filter waveform input 5kHz.

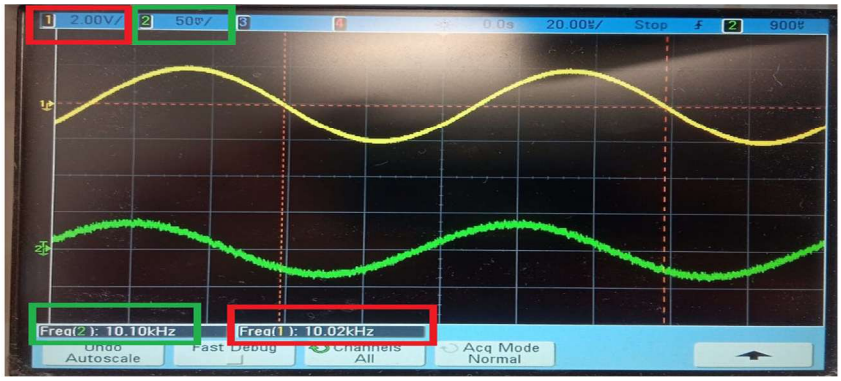

Figure 17. FIR filter waveform input 10kHz.

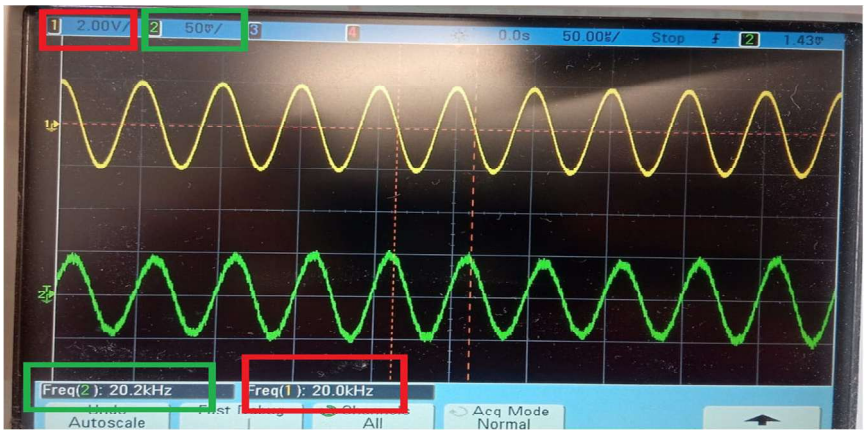

Figure 18. FIR filter waveform input 20kHz.

Figure 19. FIR filter waveform input 21kHz (Cut-off point)

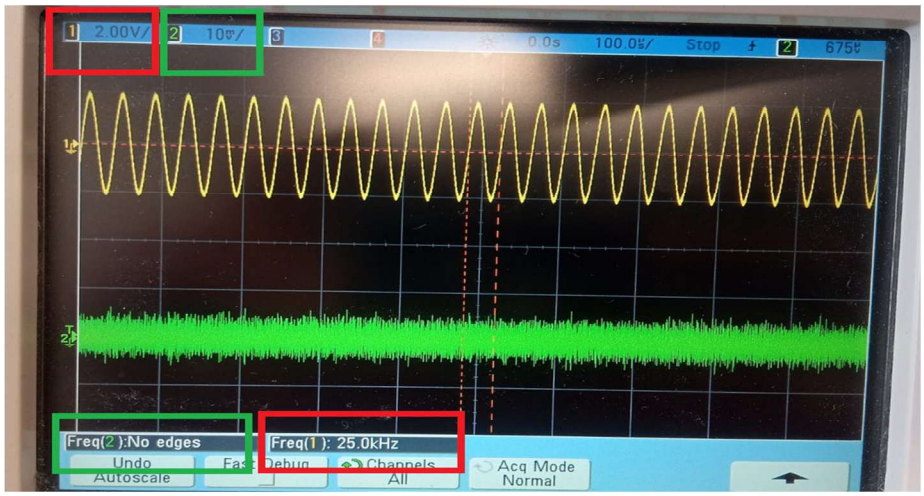

Figure 20. FIR filter waveform input 25kHz.

It is possible to plot the FIR filter frequency response graph based on amplitude and frequency measurement. See Table 7: Amplitude and Frequency Measurement of FIR Filters and Figure 21 for FIR Filter Frequency Response

FIR Filter			
Input Signal (Yellow waveform)		Output Signal (Green Waveform)	
Voltage(v)	Frequency(Hz)	Voltage(v)	Frequency(Hz)
2	1000	1	1000
2	5000	0.5	5000
2	10000	0.05	10000
2	15000	0.05	15000
2	18000	0.05	18000
2	20000	0.05	20000
2	21000	0.02	0.00208
2	22000	0.01	0.001064
2	23000	0.01	No edge
2	24000	0.01	No edge
2	25000	0.01	No edge
2	27000	0.01	No edge
2	28000	0.01	No edge
2	30000	0.01	No edge

Table 7. Amplitude and Frequency measurement of FIR filter.

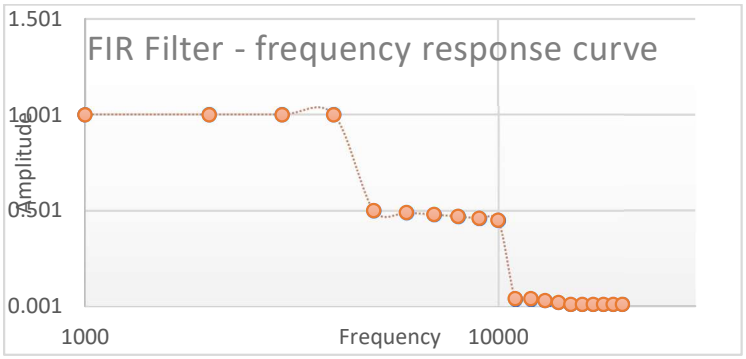

Figure 21. FIR filter frequency response.

In tracing or debugging, I used ‘’Signal Tap Logic Analyzer ‘’in FPGA development environments. It can also observe the internal signal wave generation and compare it to a real-time waveform. Moreover, it helps to analyse and observe the behaviour of signals within the FPGA design during simulation or emulation, which can aid in verification and debugging. See below figure 22 for waveform generation using the signal tap logic analyser tool.

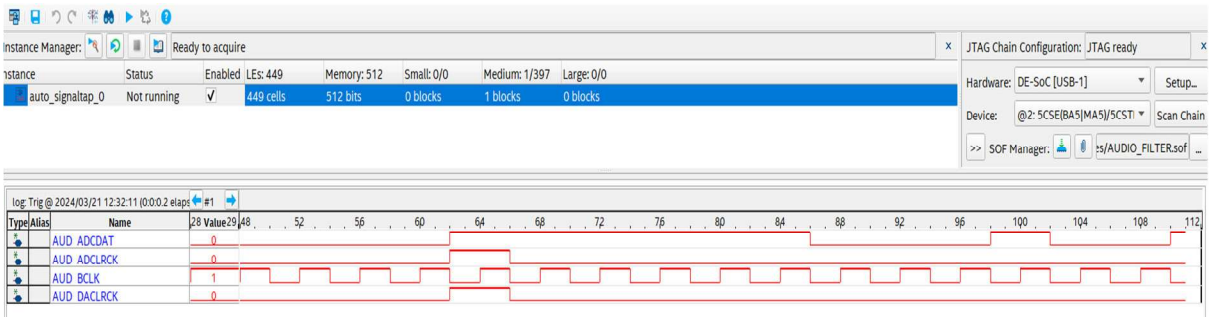

Figure 22. Waveform generation using the signal tap logic analyser tool.

Discussion

In the implementation of the FIR filter, I noticed some points for further discussion. See below for points.

1. Phase shift of input and output waveforms.
2. Filter coefficients.
3. Relationship between Frequency and Amplitude.
4. Throughputs and latency of the FIR filter.
5. Advantages of FPGA-based filters as compared to traditional filters.

I understood the phase shifts between the input and output signals in the finite impulse response (FIR) filter. It depends on the design implementation. In detail, when an input signal is applied, the filter performs a weighted sum of the current and past input samples according to the filter coefficients. These delays can occur due to several factors, such as logic delays (time taken for generating the output signal when input is applied to logic gates or processing time of logic gates), multiplier delays (filter operations may cause delays, depending on the multiplier architecture employed in the FPGA), and register delays (registers used to store prior input samples for the accumulation procedure cause a fixed delay in the signal). See Figure 23 for the for the phase shift difference between input and output signals.

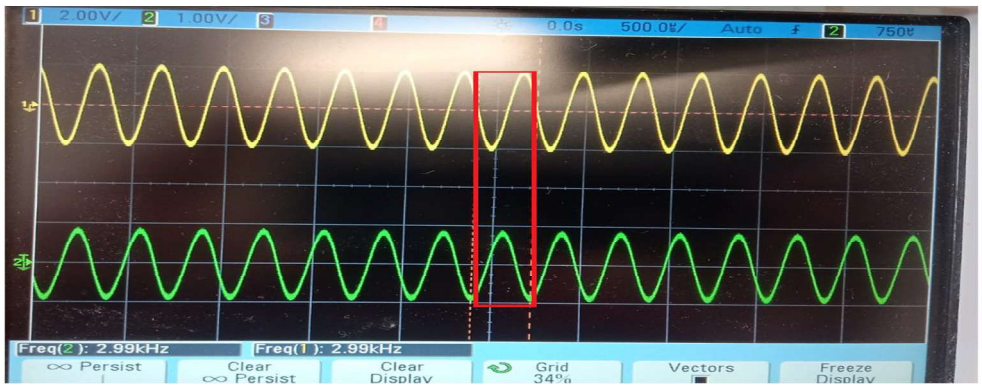

Figure 23. Phase shift difference of input and output signals.

The consequences of phase shift in FPGA FIR filters can have various effects depending on the application. For example, in audio applications, a very small phase shift may not be noticeable. However, for specific audio effects or signal reconstruction, avoid phase shift. In order, the finite impulse response filter output frequency and amplitude are related to the input frequency. When the input frequency is increased from 1 kHz to 25 kHz, the cutoff frequency is around 21 kHz. After 21 kHz, frequency and amplitude suddenly drop to feasible values. See Table 7 for the for the amplitude and frequency relations of input and output signals.

Finally, throughput in an FIR filter refers to the rate at which it can process data samples. It's typically measured in millions of samples per second (MSPS). It depends on some factors, such as the number of filters, clock speed, and architecture implementation. Latency is the time delay between an input sample entering the filter and the corresponding output sample being produced. It's also affected by some factors, such as the number of taps and overlapping computations. The advantages of a FPGA-based filter as compared to a traditional filter are high throughput, low latency, flexibility, and parallelization (parallel execution of multiple filter operations).

Conclusion

According to the objectives indicated in aims, the individual VHDL coding for the codec, S2P, and FIR filter is completed first, and their waveforms are tested under various stimulation settings using Questa-Intel FPGA Starter Edition 2023.3 or ModelSim HDL Simulator. In the codec area, I have a thorough understanding of serial and control interface (I2C and I2S) communication protocols. However, exploring serial-to-parallel and parallel-to-serial conversion with shift registers, as well as the functions of bit and word counters for data tracking, has greatly improved my understanding of their internal principles. Finally, data storage and transmission techniques are addressed, including strobe signals (ADCstb, DACstb) and ready signals (ADCrdy, DACrdy).

Secondly, acquire the hardware FPGA implementation using TeraSIC's DE1-SoC board. In this portion, I discovered how individual VHD files (Codec, S2P, and FIR) are integrated and executed in a single programme, creating a top-level priority. However, block diagram and schematic design are required, along with pin assignment and device programming. I utilised common laboratory equipment, such as a functional generator (signal generator) and an oscilloscope, to plot the frequency response curve of a low-pass FIR filter. In this section, I employed an additional function generating tool, such as sweep waveform generation from 1 kHz to 30 kHz and 2-volt peak-to-peak (Vpp), to further analyse the whole filtering operation.

Reference

1. J. Zheng and Z. Wei, "FIR Filter Design Based on FPGA," *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Changsha, China, 2018, pp. 36-40, doi: 10.1109/ICMTMA.2018.00016.
2. Panayotatos, P. (2005) Frequency Response of Filters. Rutgers University, New Brunswick.
3. Quayyum, A. and Mazher, M. (2012) Design of Programmable, Efficient Finite Impulse Response Filter Based on Distributive Arithmetic Algorithm. International Journal of Information Technology and Electrical Engineering, 1, 19-24.
4. Litwin, L. (2002) FIR and IIR Digital Filters. IEEE Potentials, 19, 28-31.
5. Manal, H.J. and Asaad, H.S. (2013) High-Pass Digital Filter Implementation Using FPGA. IEEE International Journal of Advanced Computer Science and Applications (IJACSA), 13, 41-50.
6. IEEE International Roadmap for Devices and Systems (IEEE IRDS™), version 2023 edition, January 2023. [Online]. Available: https://irds.ieee.org/
7. Monmasson, E., et al. (2011) FPGAs in Industrial Control Applications. IEEE Transactions on Industrial informatics, 7, 224-243. http://dx.doi.org/10.1109/TII.2011.2123908
8. Wenjing, H., Guoyun, Z. and Waiyun, L. (2011) Self-Programmable Multipurpose Digital Filter Design Based on FPGA. IEEE Proceedings of International Conference on Internet Technology and Applications (iTAP), Wuhan, August 2011, 1-5.
9. Wolfson Microelectronics, "WM8731 Stereo Audio Codec (WM8731L) Data Sheet," Cirrus Logic, Edinburgh, Scotland, UK, 2006. [Online]. Available: https://www.cirrus.com/en/pubs/proDatasheet/WM8731_v4.5.pdf. [Accessed: March 16, 2024].
10. Terasic Technologies. "DE1-SoC Development Board," Terasic Technologies, Hsinchu, Taiwan, 2013. [Online]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=836&PartNo=1. [Accessed: March 16, 2024].