IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141273

Implementation of real-time audio signal processing using FPGA-based digital FIR filter

Dr. S G Hiremath[1], Hemanth Kumar N N[2], Srinivasa T[3], Tabrez Pasha[4], Yogeshwar Gowda P[5]

Dept. of ECE, East West Institute of Technology, Bangalore[1-5]

Abstract: This Project paper present the implementation of digital filters on an FPGA platform for real-time audio signal processing. This utilizes the Basys-3 FPGA board and the peripheral module (PMOD) I2S2 accessory to process audio data using digital finite impulse response (FIR) filters. The extension of the filter kernel length is achieved by converting the dual- channel input to a single-channel output, and the use of free LUTs (Look Up Tables) as digital signal processing (DSP) multipliers, optimizing the utilization of available resources. The successful implementation of the filters highlighted the potential for FPGA- based solutions in audio engineering and digital signal processing. The results provide valuable insights that can guide future work in optimizing FPGA- based digital filters for audio engineering.

Keywords: Digital filter,Field Programmable gate array(FPGA),Digital Signal Processing(DSP).

I. INTRODUCTION

Digital signal processing techniques have revolutionized the field of audio engineering by enabling advanced manipulation and analysis of audio signals. Real-time processing of audio data is crucial for applications such as audio effects, noise cancellation, speech recognition, and audio enhancement. Field Programmable Gate Arrays (FPGAs) have emerged as a powerful platform for implementing real-time signal processing systems due to their reconfigurability, parallel processing capabilities, and low latency. In this paper, we present an implementation of FIR digital filters on an FPGA platform for real-time audio signal processing.

In this paper, we focus on the implementation of essential filters for audio processing, including low- pass, high-pass, band-pass, and moving average filters. These filters are widely used in various audio applications, such as frequency shaping, noise removal, and signal enhancement. We justify the choice of these filters based on their relevance to audio processing and the human auditory spectrum's frequency range.

To optimize the filter performance and resource utilization, we explore techniques to extend the filter kernel lengthWe demonstrate how the conversion from dual-channel input to single-channel output allows for a larger kernel length, maximizing the utilization of the available resources on the FPGA. Furthermore, we investigate the utilization of free Look- Up Tables (LUTs) as Digital Signal Processing (DSP) multipliers, enabling additional computational capacity without consuming extra resources.

The research presented in this paper contributes to FPGA- based audio signal processing by showing the feasibility and effectiveness of implementing digital filters on the Basys-3 FPGA board. The outcomes of our paper provide valuable insights into the design and optimization of real- time audio processing systems using FPGAs. The findings and methodologies discussed in this paper lay the groundwork for further advancements in FPGA-based audio processing and inspire future research and innovation in this domain.

II. LITERATURE REVIEW

Introduction to FPGAs Field-Programmable Gate Arrays (FPGAs) have emerged as powerful hardware platforms for implementing various digital signal processing (DSP) applications, including real-time audio processing. FPGAs are programmable semiconductor devices that consist of an array of configurable logic blocks (CLBs), embedded memory blocks (BRAM), digital signal processing blocks (DSPs), and input/output (I/O) blocks interconnected by a network of programmable routing resources. What sets FPGAs apart from other devices is their reconfigurability. Unlike ASICs, which are designed for specific functions and cannot be altered once manufactured, FPGAs can be programmed and reprogrammed to implement different functionalities, making them highly adaptable to changing requirements.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141273

The Basys3 works with Xilinx's new high- performance Vivado Design Suite. Vivado includes many new tools and design flows that facilitate and enhance the latest design methods.

III. METHODOLOGY

Hardware

In this section, the hardware components employed in the implementation of digital filters for real-time audio processing has been discussed. The paper utilizes the Basys-3 FPGA board and the PMOD I2S2 accessory, both of which play pivotal roles in enabling efficient and effective audio filtering.

Fig.1.Digilent Basys-3 FPGA Board view.

The Basys-3 FPGA board, manufactured by Digilent, serves as the primary hardware platform for the paper shown in Fig. 1. The Basys-3 board features a Xilinx Artix-7 FPGA, offering 33,280 logic cells in 5200 slices, along with 90 DSP slices, which provides ample resources for implementing complex digital signal processing algorithms, including digital filters. Additionally, it offers a range of input/output interfaces, including through PMOD accessories, allowing for seamless integration with external devices and peripherals, making it well-suited for audio processing applications.

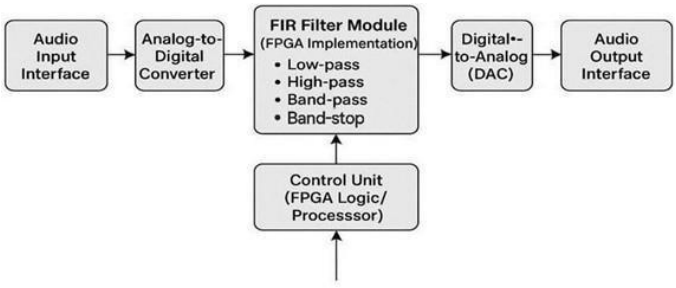

Fig.2.Block Diagram of setup

To facilitate audio data transmission into and out of the FPGA, we utilize the PMOD I2S2 accessory which is presented in Fig. 2. The PMOD I2S2 is a PMOD (peripheral module) specifically designed for audio applications. It supports the Inter-IC Sound (I2S) protocol, a widely used standard for digital audio communication. The PMOD I2S2 enables bidirectional audio data transfer between the FPGA and external audio devices, such as microphones, speakers, or audio codecs.

By leveraging the capabilities of the PMOD I2S2, we can efficiently process audio signals in real-time using the FPGA's computational power.

The successful implementation of our FPGA- based audio signal processing system relies on the integration of various other components including an audio source, software to program the FPGA, and a speaker. The audio source in our setup is a smartphone equipped with a 3.5mm audio jack, which plays back the sound to be processed. The software tool used for programming the FPGA and flashing it with the designed System Verilog code is Xilinx Vivado.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025
DOI: 10.17148/IJARCCE.2025.141273

This powerful software suite enables the configuration and control of the FPGA, facilitating the execution of our audio processing algorithms. Lastly, the Beoplay A1 speaker with a 3.5mm audio jack serves as the output device, playing back the analog audio signal generated by the FPGA. The seamless integration of these components forms a comprehensive audio processing system, enabling real-time signal manipulation, and demonstrating the practical application of FPGA technology in the audio domain.

FILTER DESIGN

When designing digital filters for audio applications, several constraints need to be considered to ensure optimal performance and compatibility with the target hardware. In the context of audio processing on a FPGA board, the following constraints are particularly relevant: the audio spectrum, the available DSP units on the FPGA, and the sampling frequency.

The human audio spectrum, which encompasses the range of audible frequencies, typically extends from 20 Hz to 20 kHz. To accurately process audio signals within this frequency range, it is essential to choose filters that provide sufficient frequency response coverage. The filters should be designed to handle frequencies from 20 Hz to 20 kHz without significant attenuation or distortion in their passband, and sufficient attenuation in their stopbands so that an average human can hear the effects of the filters.

Another important consideration is the available resources on the Basys-3 FPGA board. The board is equipped with a limited number of DSP units, which are essential for implementing digital filters efficiently. DSP units are specialized hardware blocks within the FPGA designed for digital signal processing operations. The number of available DSP units determines the kernel length of the filters implemented. In the case of the Basys-3 FPGA board, having 90 DSP units means the filter design should consider the utilization of these units to achieve the desired audio processing functionality.

Furthermore, the sampling frequency of the PMOD I2S2 module plays a crucial role in filter design. The PMOD I2S2 module provides an interface for audio input and output and operates at a sampling frequency of 44.1 kHz [8]. This sampling frequency represents the number of samples taken per second from the analog audio signal. To ensure accurate audio processing, the filters must be designed to handle the specific sampling frequency of 44.1 kHz. This involves considering the cutoff frequencies and filter characteristics that are appropriate for this sampling rate to avoid aliasing and ensure accurate reconstruction of the audio signal.

Selected Filters With regards to the constraints introduced in the previous section, the following filters were selected to be implemented. All these filters are dual channel, with a kernel length of 45 for each filter.

1. Low-pass Filter with a cut-off frequency of 1KHz.
2. High-pass Filter with a cut-off frequency of 2KHz.
3. Band-pass Filter with passband from 1KHz to 4KHz.
4. Band-stop Filter with passband from 1KHz to 4KHz. Generation of Coefficients

The coefficients for the digital filters were generated using GNU Octave, a high-level programming language for numerical computations. The “fir1” function is used to generate the filter coefficients. This function creates a finite impulse response (FIR) filter with the specified number of taps and cutoff frequencies. The resulting coefficients are then scaled to fit within the desired coefficient width of 16 bits. The code also prints out the coefficients in a formatted, human-readable format

Fig.3. PMOD I2S2 module view

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141273

Site Type	Utilization		
	Used	*Available*	*% Utilization*
Slice LUTs	383	20800	1.84%
DSPs	89	90	99%

IMPLEMENTATION

The Digilent Basys-3 FPGA is programmed in System Verilog using Xilinx Vivado 2019.2. The FIR engine is divided into two main files: firbackend.sv and fir.sv. All the code can be found in . The file firbackend.sv contains System Verilog code implements FIR filter engine capable of applying four different filters to incoming data. It utilizes buffer elements to store the input data and compute the filtered output based on the selected filter's coefficients. The coefficients for the filters are a part of this file. Based on the currently selected filter, the FIR filter engine computes the output of the filter on receiving each new packet. The output is computed using the standard formula for FIR filters, multiplying the coefficients with the corresponding buffer values, and accumulating the results.

The file fir.sv contains modules that handle and connect the AXI stream from the PMOD input to the FIR filter backend and forward the output from the FIR filter to the output stream. The file Basys-3-Master.xdc contains the mapped pins and switches between hardware and software, and the clocks used Using single-channel filter instead of dual-channel filter

By converting the dual-channel input to a single- channel output, the available resources can now be dedicated entirely to a single channel, allowing for a higher number of taps to be accommodated. By utilizing a single-channel filter, the number of taps or coefficients can be effectively doubled compared to a dual-channel filter design with the same available resources. The decision to use a single-channel filter should be based on the application's nature and the specific requirements of the signal processing task. In scenarios where the input signal contains significant information in both channels, or when preserving stereo imaging is crucial, a dual- channel filter may be more appropriate.

Site Type	Utilization		
	Used	*Available*	*% Utilization*
Slice LUTs	18948	20800	91.1%
DSPs	89	90	99%

However, when working with signals that do not necessitate dual-channel processing, employing a single- channel filter offers a practical solution for extending the filter kernel length and enhancing the filtering capabilities.

Using free LUTs as DSP Multipliers FPGAs consist of an array of LUTs that can be configured to implement any desired Boolean function. In many applications, the LUTs are not fully utilized, leaving some LUTs unused or underutilized. By repurposing these unused LUTs as DSP multipliers, additional resources can be made available for extending the filter kernel length. The process involves mapping the multiplier Digital Signal Processing (DSP) units of the FPGA to the available free LUTs. This mapping allows the FPGA to effectively use the LUTs as multipliers, augmenting the number of available multiplier blocks beyond the default count provided by the device. In our case, after mapping the multiplier DSP units to free LUTs, 73 extra multiplier blocks were obtained. After applying both the techniques to increase filter length, we could implement single- channel filters with a length of 163, which is a 262% increase over our original 45-tap dual-channel filters.

The central processing unit in the FPGA includes a control unit implemented using programmable logic or a soft processor. This control unit manages filter selection based on user inputs or predefined logic and configures the FIR filter module accordingly.

The FIR filter module is coded in SystemVerilog and supports low-pass, high-pass, and band-pass filtering. These filters are essential for shaping, cleaning, or enhancing specific audio frequencies. The digital filters operate using a multiply-accumulate (MAC) approach, where incoming audio samples are multiplied by pre-defined filter coefficients and summed to produce the filtered output.

The coefficients for each filter are generated using GNU Octave's fir1 function, which designs filters based on the desired specifications and sampling rate. These coefficients are either hardcoded or loaded dynamically into the FPGA. To increase the kernel length of the filter (i.e., the number of taps), two optimization techniques are applied:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141273

converting dual- channel audio to single-channel, and repurposing unused Look-Up Tables (LUTs) as additional DSP multipliers.

LOW PASS: A low pass filter (LPF) is an electronic or digital filter that allows low- frequency signals to pass through while attenuating or blocking high- frequency signals. It is commonly used in audio processing to remove high-frequency noise and in signal processing to smooth signals.

HIGH PASS: It allows high-frequency signals to pass while blocking low-frequency signals. High pass filters are used to remove low-frequency interference such as hum or DC offset in audio and communication systems. For instance, in an audio system, a high pass filter will suppress bass frequencies and let the treble pass, making it useful in situations where deep sounds are unwanted.

BAND PASS: A band pass filter (BPF) allows only a specific range of frequencies to pass through while blocking frequencies that are lower or higher than the desired band. This makes band pass filters extremely useful in applications like wireless communication, where only signals within a certain frequency range should be received or transmitted.

BAND STOP: A Band Stop Filter (BSF) — also known as a Band Reject Filter or Notch Filter — is an electronic filter that blocks (attenuates) frequencies within a specific range (band) and allows frequencies outside that range to pass.

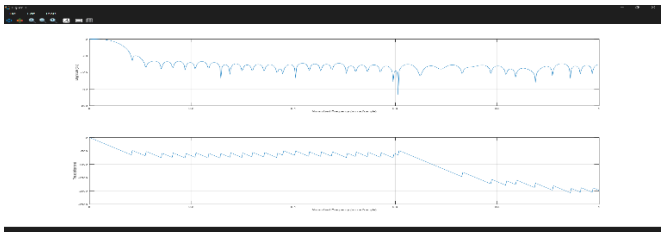

Fig 4Magnitude Response and Phase Shift of the 89-tap single-channel Low- pass filter

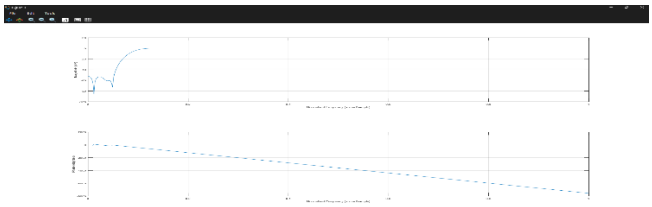

Fig. 5. Magnitude Response and Phase Shift of the 89-tap single-channel High-pass filter

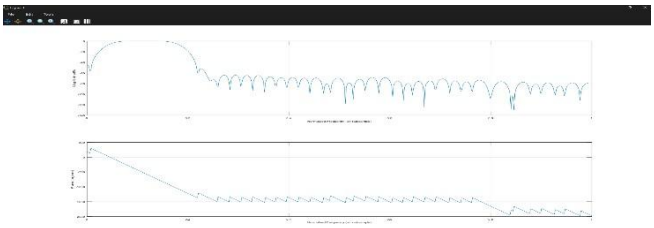

Fig. 6. Magnitude Response and Phase Shift of the 89-tap single-channel Band- pass filter

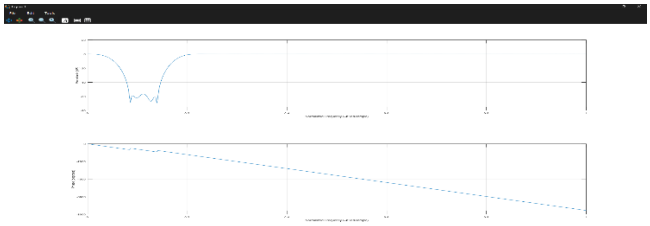

Fig. 7. Magnitude Response and Phase Shift of the 89-tap single-channel Band- stop filter

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141273

IV. RESULTS AND DISCUSSION

In this project, noise cancellation was achieved by implementing four different FIR-based filters—Low-Pass, High-Pass, Band-Pass, and Band-Stop—on the Artix-7 FPGA. Each filter was tested with real-time audio signals to evaluate how well it isolates useful speech information from background noise. The FPGA provided sufficient parallelism to perform filtering with almost zero perceptible delay.

Fig 8: The working hardware setup.

The Low-Pass Filter (LPF) successfully attenuated high- frequency noise such as sharp background sounds, electrical interference, and hissing noise. As a result, the speech waveform became smoother and clearer in the upper frequency range. The High-Pass Filter (HPF) was effective in suppressing low-frequency rumble, microphone vibrations, and environmental hum, making speech more crisp and reducing muffled components that normally interfere with clarity.

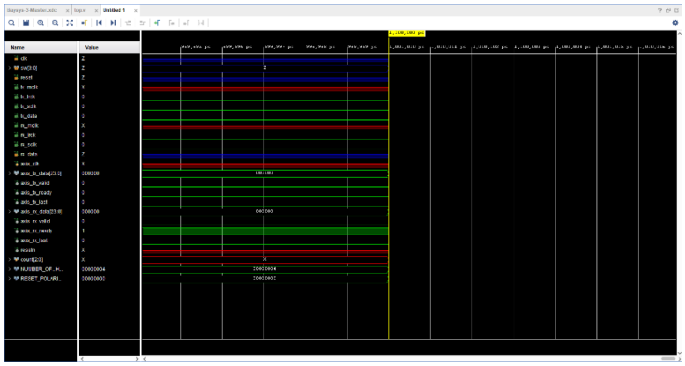

Fig 9 Simulation

When testing the Band-Pass Filter (BPF), the system produced the most natural-sounding speech. Because the passband was tuned to the main speech frequency range, unwanted components outside this band were eliminated while preserving the important harmonics of human voice. This filter gave the best balance between noise reduction and speech quality.

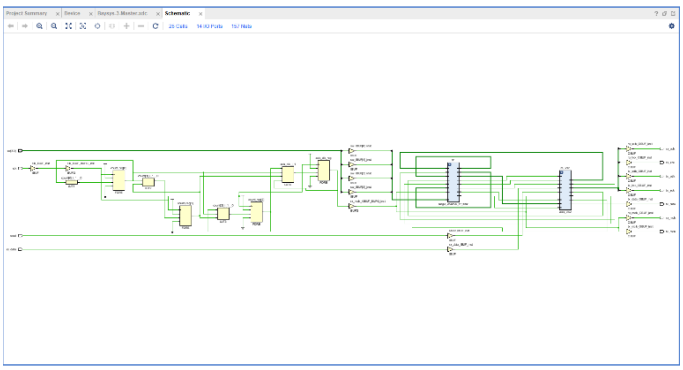

Fig 10 Schematic

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025
DOI: 10.17148/IJARCCE.2025.141273

The Band-Stop Filter (BSF), or notch filter, was particularly useful in removing specific unwanted tones such as 50/60 Hz electrical hum. During testing, the BSF provided clean suppression of these narrowband noises without disturbing the rest of the audio spectrum. This improved the stability of the signal when tonal noise sources were present.
These patterns show the distribution of LUTs, flip-flops, routing interconnect, and other synthesized components. The presence of dense vertical blue strips suggests that most of the logic is concentrated along specific columns, possibly corresponding to modules such as filters, MAC units, or control logic block.
The figure shows the Device Floorplan View of the FPGA obtained after synthesis and implementation in Xilinx Vivado. This view provides a detailed visualization of how the design has been mapped, placed, and routed on the physical FPGA fabric.
The FPGA area is divided into multiple regions, each corresponding to specific hardware resources such as Configurable Logic Blocks (CLBs), DSP slices, Block RAMs (BRAMs), I/O banks, and clock regions

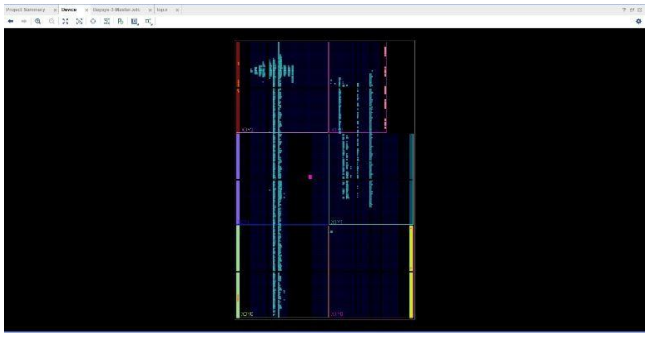

Fig 11 Implementation of Schematic

V. CONCLUSION

In conclusion, the proposed FPGA-based real-time audio enhancement and assistance system demonstrates an effective solution for improving speech clarity, reducing noise, and supporting assistive listening applications. By leveraging the parallel architecture of the Artix-7 FPGA, the system achieves fast, deterministic audio processing with minimal latency— an essential requirement for real-time auditory tasks.
The prototype successfully demonstrates clear improvement in speech clarity and background noise reduction. The processed audio output exhibits noticeably higher intelligibility even in moderately noisy environments. The low latency of FPGA-based processing ensures natural hearing without echo, making it suitable for continuous use. Optional speech- to-text conversion, performed on a companion device, accurately transcribes lecture speech and displays the text, enabling accessibility for individuals with hearing impairment.

REFERENCES

[1]. E. M. Abdulzahra, M. Al-Dulaimi, H. Wahhab, A. Amer, “Design and Implementation of Communication Digital FIR Filter for Audio Signals on the FPGA Platform,” Journal of Communications, Vol. 18, pp. 89-97, 2023 doi: 10.12720/jcm.18.2.89-96.
[2]. Abdulzahra, Mohammed & Al-Dulaimi, Mohammed & Wahhab, Husam & Amer, Ahmed. (2023). Design and Implementation of Communication Digital FIR Filter for Audio Signals on the FPGA Platform. Journal of Communications. Volume 18, 89-97. 10.12720/jcm.18.2.89-96.
[3]. Vellaiappan, Elamaran & Upadhyay, H.N. & Raju, Nallusamy & Kumaravelu, Narasimhan. (2015). Real time audio denoising using digital fir filters with FPGA implementation. International Journal of Pharmacy and Technology. 7. 9802-9810.
[4]. Shensheng Tang, Siong Moua, Yi Xie and Yi Zheng. (2022). FPGA based Implementation of an Audio Signal Processing System on Zedboard. Journal of Smart Technology Applications Vol.3, No.1 (2022), pp.1-20.
[5]. Mohammed Abdulzahra Ahmed Al-Dulaimi, Husam A. Wahhab, and Ahmed Abdulhussein Amer. (2023). Design and Implementation of Communication Digital FIR Filter for Audio Signals on the FPGA Platform. Journal of Communications vol. 18, no. 2, February 2023.
[6]. R. Smith and Y. Chang, ”Advancements in Human-Robot Interaction: A Healthcare Perspective,” IEEE Robotics & Automation Magazine, vol. 27, no. 2, pp. 85-94, 2021.
[7]. C. Ünsalan, B. Tar, “Digital system design with FPGA: Implementation using Verilog and VHDL,” McGraw-Hill Education, 2017.
[8]. Fahad Syed. “Real-Time Audio Processing on Basys-3FPGA.” github.com. https://github.com/sinandredemption/fir_basys3 accessed Jun. 29, 2023

