Sound Processing Module using HDL for FPGA

Travimadox Webb[1], **Piwani Nkomo**[2], **and Nkosinathi Ntuli**[3]

[1]**WBBTRA001**
[2]**NKMPIW001**
[3]**NTLNKO007**

ABSTRACT

This project develops a Sound Processing Module (SPM) to calculate minimum and maximum amplitude values in audio signals. The SPM offers two functionalities: basic min-max filtering for entire audio clips and interval-based analysis for specified intervals.

Keywords: SPM, HDL, Verilog, Min-max Filters

1 PROJECT OVERVIEW

1.1 Project Description

This project focuses on developing a Sound Processing Module (SPM) for calculating the minimum and maximum amplitude values within audio signals. The project has two sections:

- **Basic Min-Max Filtering:** Calculate the overall minimum and maximum amplitude values within an entire audio clip.
- **Interval-Based Min-Max Analysis:** Determine the minimum and maximum amplitude values within specified intervals of an audio clip.

The project will follow a two-phase approach:

1. **Phase 1(Simulation)** : A Sound Processing Module simulation model will be developed and verified .
2. **Phase 2(Hardware Implementation):** The project will transition to hardware implementation on a NEXYS A7 FPGA board after successful simulation.

1.2 Significance

Min-max filters form a foundation for various audio processing tasks, including noise reduction and signal shaping. They can effectively remove impulsive noises like clicks, pops, and digital glitches by replacing isolated spikes with the minimum or maximum value within a short neighbourhood. Additionally, min-max filters can smooth out compression artifacts caused by overly aggressive audio compression, reducing audible distortion. In signal shaping, they can serve as rough peak limiters to prevent audio clipping and enable creative envelope manipulation by modifying the perceived overall loudness contour of an audio signal.

1.3 Project Goals

1. Develop a Sound Processing Module for implementation on an FPGA to perform audio signal analysis and provide minimum and maximum value computations.
2. Enhance the module's functionality to calculate interval-based minimum and maximum values for user-specified interval lengths within an audio clip.
3. Ensure the module's correct operation through rigorous simulation and hardware testing.
4. Optimise the module's performance, if necessary.

1.4 Project Objectives

1.4.1 Simulation Phase

- Design and implement the Sound Processing Module's functionality using Verilog.
- Develop a comprehensive test bench to simulate the module's behaviour with various input audio clips and interval lengths.
- Verify the module's output by analysing simulation results and ensuring compliance with specified requirements.

1.4.2 Hardware Implementation Phase

- Synthesise the HDL code for the Sound Processing Module, targeting the NEXYS FPGA board.
- Integrate the module with the FPGA board's audio input/output interfaces, ensuring proper connectivity and data transfer.
- Validate the module's functionality on the FPGA board by testing it with real-world and interval lengths and audio clips.
- Explore and implement optimisation techniques, such as pipelining and parallelism, to enhance the module's performance, if necessary.

1.4.3 Documentation and Reporting

- Maintain comprehensive documentation throughout the project's lifecycle, including design specifications, simulation results, and hardware implementation details.
- Prepare a detailed final report highlighting the project's objectives, methodologies, results, and any significant findings or challenges encountered during development.

2 PROPOSED SOLUTION

2.1 System Design & Implementation Overview

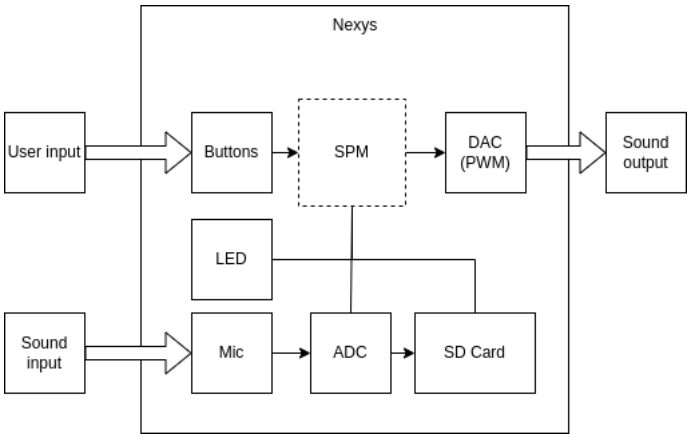
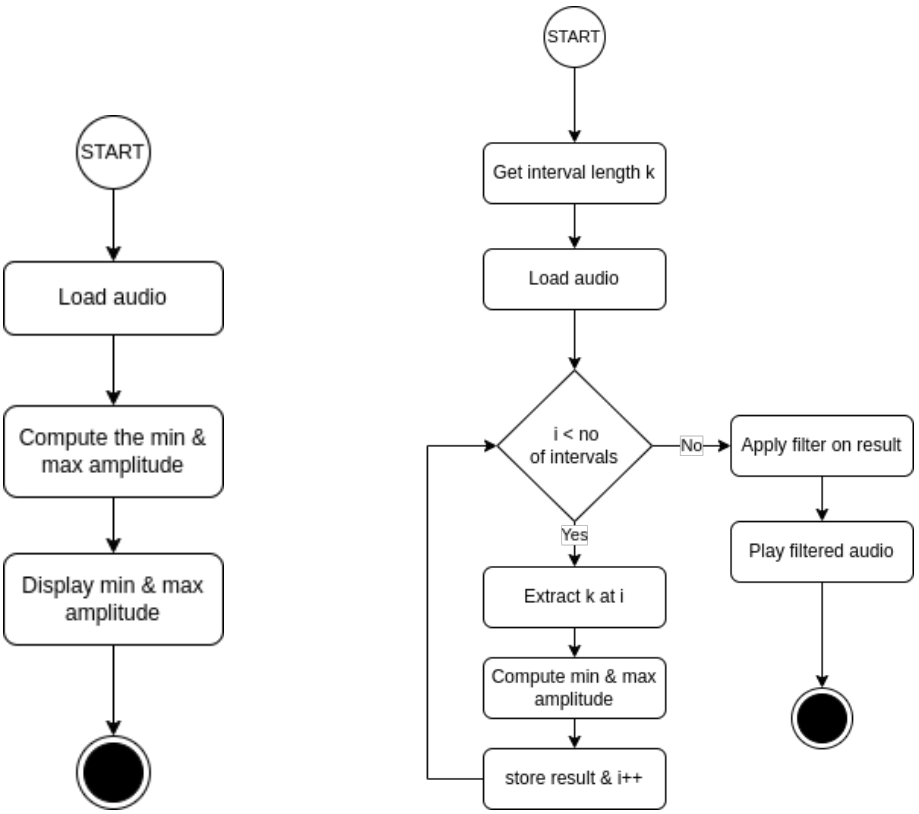


Figure 1. The SPM system block diagram

The SPM accelerator will be developed and simulated using Vivado and prototyped on the NEXYS 4 board. The system block diagram is shown in Fig 1. The solution will implemented over a two-phase approach.

The solution for the first phase is simulated using Verilog. An audio signal will be loaded into the Verilog testbench and the module will compute the minimum and maximum amplitude of the audio and output the results on the terminal. Fig 2b shows a flowchart of the implementation.
The second phase solution is implemented on a Nexys board. An audio can be recorded using an onboard microphone and stored on an SD card, or a pre-recorded audio can be loaded onto the same SD card.



(a) Phase 1 implementation **(b)** Phase 2 implementation

Figure 2. Phase 1 & Phase 2

For the recording option, the user will initiate the process by pressing a dedicated record button on the board, triggering the activation of a red LED to indicate recording has started. A subsequent button press will stop recording, with the LED turning off to signal the end of the process. The recorded audio is saved as a WAV on an SD card attached to the board. Following successful storage of the audio on the SD card, the user will be prompted through a display on the board to enter a desired interval length for filtering. However, a preset value will be used instead due to the absence of a numerical input module. Subsequently, the audio will undergo processing through the designated filter, with the filtered audio saved onto the SD card and played back through an audio output.

The user will initiate the process for the pre-recorded audio option by pressing a dedicated play button. In this case, the user will be prompted to enter the interval after that the pre-recorded will be loaded from the SD card, filtered and saved. The filtered audio will then be played through the audio output. Fig 2a shows a flowchart of the implementation.

2.2 Algorithm Analysis

Several algorithms can be used to implement the min-max filters for the audio signals. The comparison between popular algorithms is done in the Table 1 for basic min-max filter and Table 2 or the interval-based min-max filter. Afterwhich, the most appropriate algorithm for implementation on a Nexys A7 FPGA board is chosen, along with justification.

Algorithm	**Description**	**Time Complexity**	**Space Complexity**
Linear Search	Simple loop-based iteration, comparing each element	O(N)	O(1)
Divide & Conquer	Divides the input array into smaller subarrays, finds the minimum and maximum values in each subarray, and then combines the results.	O(N log N)	O(log N)
Tournament Method	Create a min/max tournament-like tree structure	O(N)	O(N)
Bitonic network	Constructs a bitonic network (a parallel sorting network) to find the minimum and maximum values simultaneously.	$O(\log^2 N)$	O(N)

Table 1. Basic Min-Max Filter Algorithms Analysis

Algorithm	**Description**	**Time Complexity**	**Space Complexity**
Linear Search	Iterates through the input array, dividing it into intervals of size K, and finds the minimum and maximum values for each interval.	O(N)	$O\left(\frac{N}{K}\right)$
Divide & Conquer	Divides the input array into subarrays of size K, finds the minimum and maximum values in each subarray, and then combines the results.	$O(N \log K)$	$O\left(\frac{N}{K} + K \log K\right)$
Naive Sliding Window	Calculate min/max directly for each interval	O(N * K)	O(1)
Deque-Based	Maintain potential min/max candidates in a deque, sliding it over the data	O(N)	O(K)
Hierarchical	Pre-compute min/max for blocks, use them for larger intervals	O(N) (	O(N)
Parallel Prefix Scan	Performs a parallel prefix scan on the input array to compute the minimum and maximum values for each interval efficiently.	$O\left(\frac{N}{K} + K \log K\right)$	$O\left(\frac{N}{K} + K\right)$

Table 2. Interval Based Min-Max Filter Algorithms Analysis

For implementation on the Nexys A7 FPGA board, the most appropriate algorithm would be the Linear Search algorithm for both the basic min-max filter and the interval-based min-max filter for the reasons in Table 3 :

Justification	Explanation
Simplicity	The Linear Search algorithm is relatively simple to implement, making it easier to design and verify on an FPGA board.
Resource Utilization	FPGAs have limited resources compared to general-purpose processors. The Linear Search algorithm has a low space complexity, requiring only a constant amount of extra memory for storing the minimum and maximum values. This makes it more suitable for FPGA implementation, where resources are constrained.
Parallelism	While the Linear Search algorithm is not inherently parallel, it can be easily parallelized on an FPGA by processing multiple samples simultaneously. This can be achieved by replicating the logic for finding the minimum and maximum values and processing multiple samples in parallel.
Performance	For moderately sized input arrays, which is typically the case for audio signals, the Linear Search algorithm can provide acceptable performance on an FPGA.

Table 3. Linear Search Algorithm Justification

3 PROTOTYPE SPECIFICATION

3.1 Plan of Development

3.1.1 Simulation Phase

1. Design and Implement SPM in Verilog.
2. Develop a comprehensive test bench in Verilog to simulate the SPM's behaviour with various input audio clips in WAV format, generate test vectors, and compare the SPM's output against expected results.
3. Simulate the Verilog code for the SPM and the test bench using Xilinx Vivado Simulator.
4. Analyze simulation results to verify the SPM's correct operation and compliance with specified project objectives.
5. Iteratively refine the Verilog code to address any discrepancies or issues identified during the simulation phase.

3.1.2 Hardware Implementation Phase

1. Synthesize the verified Verilog code for the SPM, targeting the Artix-7 FPGA using the Xilinx Vivado Design Suite.
2. Implement a dedicated PDM decoder module in Verilog to convert the PDM data from the Nexys A7 board's microphone into a parallel format suitable for processing by the SPM.
3. Encode the SPM's output (minimum and maximum amplitude values) into a PWM (Pulse-Width Modulation) signal and send it to the PWM audio output interface on the board.
4. Program the synthesised bitstream onto the Artix-7 FPGA on the Nexys A7 board and integrate the SPM with the audio input/output interfaces.
5. Validate the SPM's functionality on the FPGA board by testing it with real-world audio clips recorded directly from the microphone and various interval lengths, monitoring and verifying the computed minimum and maximum amplitude values against expected results.
6. Store the minimum and maximum amplitude values computed by the SPM in a CSV file on the microSD card inserted into the board's microSD card slot.
7. If required, Explore and implement optimisation techniques such as pipelining and parallelism to enhance the SPM's performance on the FPGA.

3.2 Specifications

3.2.1 Simulation Phase

The specifications for the simulation phase are shown in Table 4 below:

Specification	Name	Detail/Explanation
SP01	HDL Choice	Verilog
SP02	Simulation Environment	Xilinx Vivado Simulator
SP03	Input Audio Format	Restricted to WAV files only
SP04	Audio Length	10 seconds to 30 seconds (480,000 to 1,440,000 samples at 48 kHz sampling rate)
SP05	Audio Channel	Mono audio only (stereo audio files will be converted to mono)
SP06	Audio Data Type	32-bit floating-point format

Table 4. Simulation Phase Specifications

3.2.2 Hardware Implementation Phase

The specifications for the simulation phase are shown in Table 5 below:

Specification	Name	Detail/Explanation
SP01	FPGA Selection	Nexys A7 FPGA board from Digilent (Artix-7 FPGA)
SP02	HDL Synthesis	Xilinx Vivado Design Suite
SP03	Audio Input	PDM microphone on the Nexys A7 board, with a dedicated PDM decoder module implemented in Verilog
SP04	Audio Output	PWM (Pulse-Width Modulation) signal encoding the SPM's output
SP05	Output Storage	CSV file on the microSD card inserted into the board's microSD card slot

Table 5. Hardware Phase Specifications

REFERENCES

Nexys A7 Reference Manual - Digilent Reference.

Nexys Video DMA Audio Demo - Digilent Reference.

Douglas, S. (1996). Running max/min calculation using a pruned ordered list. *IEEE Transactions on Signal Processing*, 44(11):2872–2877. Conference Name: IEEE Transactions on Signal Processing.

Li, M., Liang, H., Liu, S., Poon, C. K., and Yuan, H. (2018). Asymptotically Optimal Algorithms for Running Max and Min Filters on Random Inputs. *IEEE Transactions on Signal Processing*, 66(13):3421–3435. Conference Name: IEEE Transactions on Signal Processing.

Nagahara, M. (2011). Min-Max Design of FIR Digital Filters by Semidefinite Programming. In Cuadrado-Laborde, C., editor, *Applications of Digital Signal Processing*. InTech.