Technical notes on using Analog Devices products and development tools
Visit our Web resources https://www.analog.com/ee-notes and https://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Evaluating SHARC-FX Performance with EEMBC's AudioMark Benchmark

Contributed by Aradhita Sharma, Li Liu, John Redford *Rev 1 –October 13, 2025*

Introduction

This EE-Note focuses on the performance of the SHARC-FX® digital signal processing (DSP) core when running the EEMBC's AudioMark benchmark[1], an open-source benchmark suite designed to simulate real-world audio pipelines including beamforming, noise suppression, and neural network-based keyword spotting.

The process of porting and optimizing AudioMark for the SHARC-FX DSP core is described in the note, leveraging its single-instruction, multiple-data (SIMD) architecture and DSP capabilities. This EE-Note walks through the setup, build, and execution steps, and presents a comparative analysis of benchmark results across different microcontroller platforms.

Overview of the SHARC-FX Core and Processors

The SHARC-FX core is a high performance DSP platform developed through a collaboration between Analog Devices and Cadence, leveraging Cadence's Xtensa® customizable processor technology. It represents a significant evolution from the earlier SHARC and SHARC+ families, offering modern architecture tailored for compute-intensive, real-time applications. The SHARC-FX DSP core brings a new level of flexibility and performance through its 4-way very long instruction word (VLIW) architecture and 256-bit SIMD engine. These features allow the processor to issue up to four operations per cycle, enabling highly parallel execution of signal processing tasks. The SIMD unit supports lane widths of 8, 16, 32, or 64 bits. This allows the processor to process multiple data in parallel, such as eight 32-bit multiply-accumulate operations per cycle. The SHARC-FX processor supports a broad range of data types, including 8- to 64-bit integers, fixed-point formats, and both 32- and 64-bit floating-point numbers, in real and complex forms. This versatility makes the SHARC-FX processor suitable for a wide array of signal processing algorithms, from signal filtering to complex machine learning inference.

In addition to their computational capabilities, SHARC-FX processors include a robust memory subsystem designed for high-throughput and low-latency access. It features 256 kB of data cache, 512 kB of tightly coupled data RAM, 32 kB of instruction cache, and 64 kB of instruction RAM. These architectural features ensure that the processors can sustain high data rates and deterministic performance, which are essential for real-time audio processing.

SHARC-FX processors are available across multiple hardware platforms, including the ADSP-21834/21835/21836/21837 and ADSP-SC834/SC835 processors[2]. For this benchmarking project, the ADSP-21835 evaluation board is utilized, which integrates the SHARC-FX core and provides a suitable platform for evaluating real-time audio processing performance using the AudioMark benchmark.

Understanding AudioMark Benchmarking

AudioMark is a modern benchmarking suite developed by the Embedded Microprocessor Benchmark Consortium (EEMBC) to evaluate the performance of embedded processors in real-world audio processing scenarios. Unlike synthetic benchmarks that focus on isolated operations, AudioMark simulates a complete audio pipeline that mirrors the signal flow found in voice-enabled edge devices. It is designed to benchmark both the DSP and machine learning (ML) capabilities of a processor.

Pipeline Components

The key components of AudioMark include:

- **Spectrum decomposition analysis**: The first stage after audio capture. It involves converting the signal from the time domain to the frequency domain to analyze its spectral characteristics.
- **Direction of arrival (DOA)**: DOA estimation determines the angle or direction from which a sound source is arriving relative to the microphone array. The system can estimate where the speaker is located, which helps in focusing the audio processing pipeline on the desired source.
- **Beamforming**: A spatial filtering technique that uses multiple microphones to enhance signals coming from a specific direction while suppressing noise and interference from others.
- **Acoustic echo cancellation (AEC)**: Removes echoes caused by the playback of the device's own audio (for example, music, voice prompts) that are picked up by the microphone. This is especially important in full-duplex systems like smart speakers or conferencing devices.
- **Single-channel noise-suppression (SCNS)**: Reduces background noise from the captured audio signal using statistical or machine learning-based algorithms. SCNS operates on a single audio channel and focuses on distinguishing speech from noise based on spectral and temporal characteristics.

- **Feature extraction**: The cleaned audio signal is transformed into a compact and informative representation suitable for machine learning. The most common technique is mel-frequency cepstral coefficients (MFCCs), which capture the spectral envelope of speech. This step reduces the dimensionality of the data while preserving the key features for classification.
- **Neural net classification**: The final stage involves feeding the extracted features into a convolutional neural network (CNN) to perform keyword spotting (KWS). The model is trained to recognize specific phrases and outputs a confidence score indicating whether the keyword is present.

Benchmark Metrics

To ensure consistency and fairness across platforms, AudioMark enforces strict rules on how the benchmark must be executed and how scores are calculated. For a benchmark score to be considered valid, the implementation must pass a series of regression tests to verify the correctness of the following core components: ABF (adaptive beamforming), AEC, ANR (automatic noise reduction), MFCC, and KWS.

The AudioMark score is calculated using the following formula:

$$\text{AudioMark Score} = \frac{\text{Iterations per second x } 1000}{1.5}$$

This formula normalizes the performance based on a fixed workload duration (1.5 seconds), allowing for consistent comparisons across devices. The benchmark is typically run for a fixed number of iterations, and the total time taken is used to compute the iterations per second. To account for differences in processor clock speeds, a secondary metric called AudioMark/MHz is also used:

$$\text{AudioMark/MHz} = \frac{\text{AudioMark Score}}{\text{Highest Core Frequency (MHz)}}$$

Optimizing AudioMark for SHARC-FX Processors

Pre-optimization Preparation

While AudioMark is designed to be portable across platforms, achieving optimal performance on SHARC-FX processors requires a combination of toolchain configuration, memory layout optimization, and low-level code optimization.

The first step is setting up the development environment using Analog Devices' CrossCore Embedded Studio® [3] (CCES) and the evaluation board SDK. The AudioMark source code is integrated into a new CCES project, and platform-specific configurations are applied, including memory mapping, cache settings, and linker script adjustments.

CMSIS libs Optimization

CMSIS NN and DSP libs are heavily engaged in AudioMark. To reflect the SHARC-FX processor capabilities in ML, a wide range of CMSIS functions are re-implemented and optimized using SHARC-FX compiler intrinsics, ensuring that each function takes full advantage of the processor's 256-bit SIMD compute unit. Further optimization techniques for SHARC-FX processors can be found in the SHARC-FX processor optimization application note [4]. This CMSIS for SHARC-FX layer not only accelerates the AudioMark benchmark but also serves as a reusable, scalable foundation for future SHARC-FX-based DSP and AI applications. This CMSIS for SHARC-FX implementation includes optimized versions of functions across multiple CMSIS modules mentioned in Table 1.

Table 1. Optimized DSP and NN Functions for the SHARC-FX Processor

NN and DSP Functions			Cycles/Call Optimization	
Module	**Function**	**Description**	**Before**	**After**
Convolve	adi_sharcfx_convolve_s8	Performs 2D convolution on quantized data.	90.46	58.75
	adi_sharcfx_depthwise_conv_3x3_s8	Executes 3x3 depthwise convolution on quantized input.	725.61	19.26
	adi_sharcfx_nn_mat_mult_kernel_s8_s16	Multiplies 8-bit and 16-bit matrices.	5.84	1.82
NN Support	adi_sharcfx_nn_mat_mult_nt_t_s8	Matrix multiplication with transposed inputs for quantized data.	324.59	55.02
	adi_sharcfx_nn_vec_mat_mult_t_s8	Vector-matrix multiplication for quantized data.	0.74	0.48
Pooling	adi_sharcfx_avgpool_s8	Applies average pooling to quantized input.	73.43	1.43
Basic Math	adi_sharcfx_add_f32	Performs element-wise addition.	0.24	0.02
	adi_sharcfx_sub_f32	Performs element-wise subtraction.	0.13	0.03
	adi_sharcfx_mult_f32	Multiplies two float arrays elementwise.	0.53	0.35
	adi_sharcfx_offset_f32	Adds a constant offset to each element.	0.22	0.03
	adi_sharcfx_scale_f32	Scales each element by a constant factor.	0.13	0.03

Complex Math	adi_sharcfx_cmplx_mag_f32	Calculates magnitude of complex numbers.	7.21	0.03
	adi_sharcfx_cmplx_mult_cmplx_f32	Multiplies two complex vectors elementwise.	0.17	0.02
Fast Math	adi_sharcfx_vlog_f32	Computes natural logarithm of each element.	0.22	0.03
Matrix	adi_sharcfx_mat_vec_mult_f32	Performs matrix-vector multiplication.	0.52	0.35
Transform	adi_sharcfx_cfft_f32	Performs complex FFT transformations.	1.59	0.03
	adi_sharcfx_rfft_fast_f32	Perform real FFT transformations.	1.34	0.04

SpeexDSP Optimization

SpeexDSP library was optimized using SHARC-FX compiler intrinsics, focused on maximizing the use of the processor's 256-bit SIMD compute unit, resulting in significant performance gains across core SpeexDSP modules such as echo cancellation and preprocessor. See Table 2.

Table 2. Optimized SpeexDSP Functions for the SHARC-FX Processor

SpeexDSP Functions			**Cycles/Call Optimization**	
Module	**Function**	**Description**	**Before**	**After**
MDF	speex_echo_cancellation	Performs echo cancellation on a frame, using adaptive filtering.	139.55	42.24
Preprocess	speex_preprocess_run	Preprocess an audio frame by applying noise suppression, voice activity detection, and other enhancements.	146.95	11.84

The result of these efforts is a highly efficient implementation of AudioMark on SHARC-FX processors that not only passes all required regression tests but also delivers strong benchmark scores.

Build and Execution Steps

This section outlines the steps required to build and run the AudioMark benchmark on the SHARC-FX processor using both command-line tools and the CCES IDE. It includes prerequisites, project structure, and detailed instructions for both workflows.

Prerequisites

Hardware:

- EV-21835-SOM
- ICE-1000 or ICE-2000 emulator

Software:

- CCES version 3.0.2 or higher
- MSYS2 shell for command-line builds to run the provided `project_build.bat` and `project_run.bat`

Cloning the Project

- Original AudioMark source from EEMBC can be found here: https://github.com/eembc/audiomark/tree/main
- SHARC-FX AudioMark project from Analog Devices can be found here: https://github.com/analogdevicesinc/audiomark-sharcfx-optimizations

Project Structure Overview

The project is organized into the following key folders:

- Audiomark/
  Contains the original AudioMark source code, SHARC-FX-specific port (ports/adi_sharcfx), and optimized DSP overrides (Lib/speexdsp/libspeexdsp)
- Audiomark-makefile/
  Used for command-line builds and execution
- Audiomark-sharc-fx-cces/
  Project files for building within the CCES IDE
- Audiomark-cces-library/
  Builds AudioMark as a static library (`.a`) for use in test case projects
- Audiomark-testcases/
  Contains five test case projects: ABF, AEC, ANR, MFCC, and KWS

Building and Running via Command Line

1. Open MSYS2 and navigate to the AudioMark-makefile directory.
2. Run the build script: `project_build.bat`.
   - The script compiles all project components, test cases, and generated `.dxe` executables for the SHARC-FX processor.
3. Ensure the SHARC-FX board is connected via ICE-1000/2000.
4. Run the build script: `project_run.bat`.
   - This loads the `.dxe` files onto the hardware and executes the benchmarking. Results are saved to `final_output.log` file.

Building and Running via CCES

1. Launch CCES and go to File > Open projects from file system, and import:
   - Audiomark-sharc-fx-cces.
   - All test cases projects from Audiomark-testcases/ (ABF, AEC, ANR, MFCC, KWS).
   - Import Audiomark-cces-library (used as a static library for the test cases).
2. Set Release mode for each project as active build configuration and build each project to generate `.dxe` files.
3. Connect the SHARC-FX board via an emulator.
4. Go to Run > Debug Configurations.
5. Create a new configuration:
   - Type: Application with GDB and OpenOCD emulator
   - Target: Analog Devices ADSP-21835W-EV-SOM
   - Speed: Max 5 MHz
6. Set the `.dxe` file under C/C++ Application.
7. Click Debug to load the program.
8. Click Resume to start execution.
9. Monitor the console for benchmark output and test case results for each imported project.

Benchmark Results

After optimization, the SHARC-FX processor achieved an AudioMark score of 14,425, or 14.42 AudioMark/MHz. Table 3 includes published results from the official EEMBC AudioMark score page with the SHARC-FX result added for comparison.

Table 3. AudioMark Efficiency Comparison Across DSP Architectures

Platform	MHz	AudioMark	AudioMark/MHz
Arm Cortex-M4 (FPGA)	25	19.02	0.76
Arm Cortex-M33 (FPGA)	20	17.66	0.88
Arm Cortex-M7 (FPGA)	25	36.98	1.48
Arm Cortex-M55 (FPGA, single run)	32	116.03	3.63
Arm Cortex-M85 (FPGA)	25	102.93	4.12
Arm Cortex-M55, U55 (FPGA, multi run)	32	508.07	15.88
SHARC-FX (ADSP-21835)	1000	14425	14.42

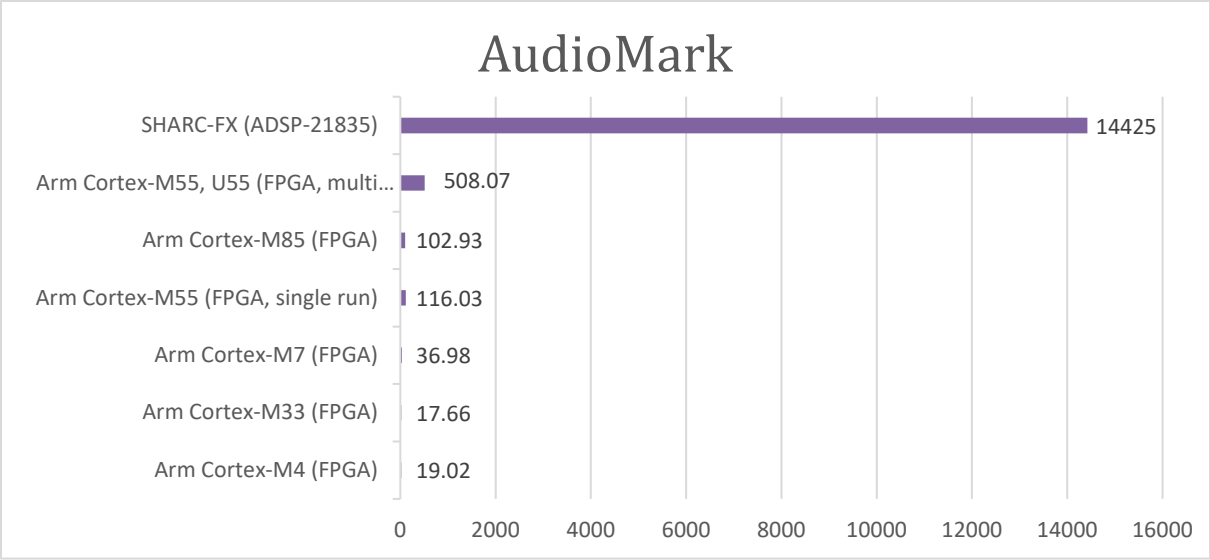

Figure 1. SHARC-FX DSP Core Performance in AudioMark Efficiency

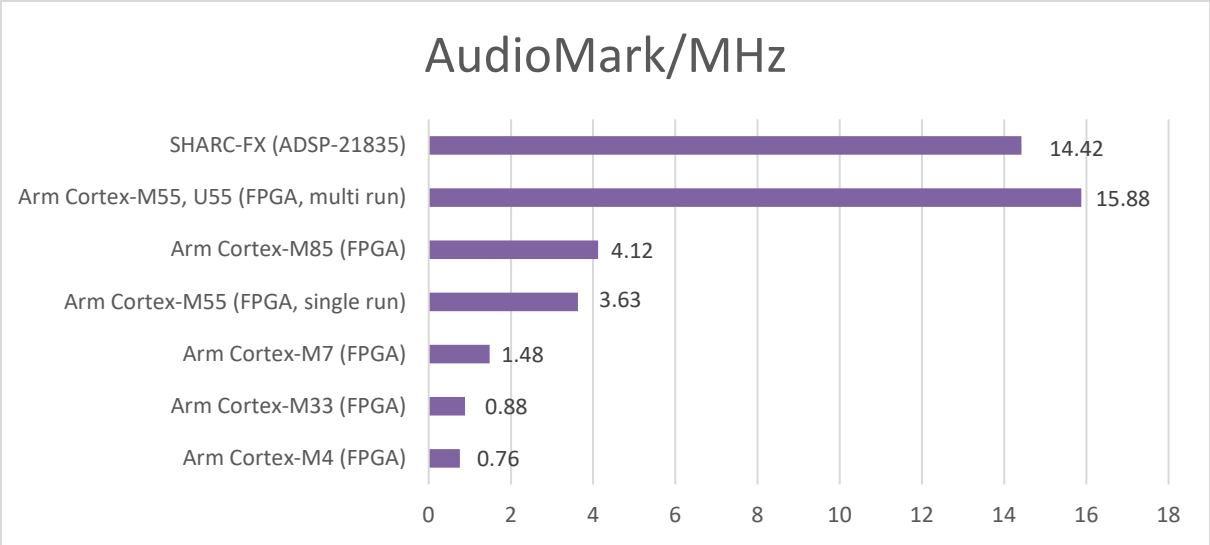

Figure 2. SHARC-FX DSP Core Performance in AudioMark per MHz Efficiency

At **14.42 AudioMark/MHz** (Figure 2), the SHARC-FX DSP core ranks just below the **Cortex-M55/U55 (15.88 AudioMark/MHz)**, one of Arm's most optimized cores for ML. With clock speeds up to **1 GHz**, the SHARC-FX DSP core demonstrates a strong and scalable capability in traditional audio and ML applications.

The results highlight the processor's ability to manage complex, real-time audio pipelines and emphasize the importance of architecture-aware tuning when deploying AI and DSP workloads on embedded platforms.

References

[1] *EEMBC AudioMark GitHub Repository: https://github.com/eembc/audiomark*

[2] *ADSP-21834/21835/21836/21837/ADSP-SC834/SC835 High Performance SHARC-FX DSP Core With Arm-Based Connectivity Data Sheet, Rev. A, August 2025, Analog Devices, Inc.*

[3] *CrossCore Embedded Studio for SHARC+ and SHARC-FX – Release, Rev. 3.0.3, Analog Devices, Inc.*

[4] *Introduction to SHARC-FX Optimization (EE-472), Rev 1, April 2025, Analog Devices, Inc.*

Document History

Revision	Description
Rev 1 – October 13, 2025 by A. Sharma	Initial Release