Portable Music Players have become immensely popular in the recent years. The success of these music players can be attributed to various factors ranging from availability of massive amount of compressed music content, complemented by declining cost for memory storage, ease of use, and easy access to online music content of different genres making them popular. The expectations from the next generation of music players are even more demanding with listeners wishing for longer audio playback time coupled with enriching listening experience.

Cortex-M series, the new generation of low cost microcontrollers from ARM®, are low power by design. Cortex™-M3 core and the recently launched Cortex™-M4 core are based on Harvard architecture with a 3-stage pipeline and implement Thumb®-2 Instruction Set Architecture (ISA) to ensure lower memory requirements. But are these MCUs capable of audio processing? Can they match the expectation of better listening experience?

To analyze the suitability of these processors for audio, first an overview of the implementation of audio processing modules commonly used in audio components like audio decoders and post processing will be provided, with examples from MP3 decoder and Equalizer. These modules will be then categorized based on the requirements they place on a processor’s Instruction Set Architecture for an efficient implementation.

The features of the Cortex-M3 and Cortex-M4 cores will be discussed with emphasis on the advantages for audio processing from an instruction set perspective. To illustrate, few code examples of modules from different audio components are discussed followed by an analysis of the cycle and instruction advantage on these processors. Finally, the typical performance numbers for popular audio codecs and audio post processing components will be shown to prove the audio capability of these processor cores.

Audio Codecs
The huge amount of compressed audio data available on the internet necessitate the need for supporting many popular audio decoders in the music player. The processing in the audio codecs involves a set of functional blocks which will be reviewed next.

Audio Encoder: The goal of an audio codec is to achieve compression of audio data while striving for perceptually transparent quality given the bitrate constraint. A diagram of a typical encoder is shown in Figure 1. Broadly speaking, compression in an audio encoder is achieved in three stages as described:

- First, the audio data to be encoded is conditioned by filtering the data. Then the audio data is framed by windowing and overlapping to enable further processing.
- Next, this data is transformed to frequency domain to exploit the redundancy in time domain. Using psychoacoustic principles, the amount of inaudible quantization noise that can be introduced is calculated.
- Finally, this data is quantized and further losslessly compressed through entropy encoding. This encoded data is packed compactly into a bitstream. Though the compression achieved is lossy, it is perceptually transparent.

Audio Decoder: The reverse of the process discussed for audio encoder is applied at the audio decoder to generate the audio data back from the bitstream. As an example, the MP3 decoder is discussed here and the high level module breakup is shown in Figure 2. The block diagram takes compressed audio in the form of MP3 file as input and outputs uncompressed audio in PCM audio.

Functional Blocks of Audio Processing Modules
To enable an analysis on the requirements for audio processing, a discussion on the functional blocks involved in audio processing modules, an audio codec and an audio post processing component, will be taken up now. The modules in the block diagrams featured in this section are coloured to represent 3 different categories.

- Green represents Multiply and Accumulate (MAC) Intensive modules
- Red represents Mix of MAC and Control code modules
- Blue represents modules which are Control code in nature
Audio Post Processing
Post processing of audio data is necessary in two scenarios. First, it is required for providing an enriching listening experience. For instance, a spatializer like stereo widening, can be used to eliminate the listening fatigue while listening over a headphone. Post processing is also needed even for basic audio playback. For instance, resampling the audio data might be required in a music player whose DAC’s supports a limited set of sampling rates.

Figure 3: Block diagram of Equalizer
As an example, an equalizer is considered to discuss the typical modules involved in post processing. An equalizer is typically applied to suit the listener’s preference, while it can also be used for correcting the response of headphones or speakers. The high level module breakup is shown in Figure 3 for a basic parametric equalizer. The modules listed below are used for modifying only a section of audio spectrum by giving a boost or a cut with controllable parameters for gain value, bandwidth, and center/cutoff frequency. Multiple such biquad IIR filters can be cascaded in series to achieve a desired response.

- Low Frequency (LF) Shelving Filter – A filter used to modify the bass section of audio spectrum
- Peaking Filter – A filter used to modify the mid range section of audio spectrum
- High Frequency (HF) Shelving Filter – A filter used to modify the treble section of audio spectrum

Categories of Audio Processing Modules
The audio processing modules can be broadly classified into three categories based on their traits. Each category of modules and their traits will be discussed in detail, and the processor requirements for their efficient implementation will be analyzed. The summary of processor requirements for audio processing discussed in this section is tabulated in Table 1.

### MAC Intensive modules
These modules are computationally intensive and are characterized by Multiply and Accumulate (MAC) operations and usually have a predetermined code flow. Signal processing operations like filtering (FIR, IIR filters), windowing, correlation, and convolution are typical examples of this category. The processor requirements for this category of modules are discussed next.

- **High precision MAC**: A high precision multiply and accumulate would be desirable for proper functioning of these modules. For instance the stability of an IIR filter requires high precision operations. The fidelity and accuracy of the output depends directly on the precision of the MAC instructions. Though high precision MAC instructions, for e.g. 32-bit multiplied by 32-bit accumulating in 64-bit result, can be simulated using lower precision instructions, it is desirable to have high precision MAC instructions for an efficient implementation.

- **SIMD capability**: The operations performed in these modules are repeated execution of few instructions yielding a compact inner loop kernel. Single Instruction Multiple Data (SIMD) capability is ideally suited for efficient execution of such modules. A processor with this feature is a added benefit for processing such modules.

- **Saturation Arithmetic**: Certain operations require the use of saturated arithmetic. For instance, when generating window coefficients at runtime or during creation of PCM output data to a desired bitwidth. The need for saturation arithmetic can be reduced, but cannot be avoided altogether due to the higher bitwidth of intermediate result. So instructions supporting saturated arithmetic are desirable for efficient implementation.

- **Branch speculation**: These modules have a predetermined code flow. Since the operations performed in these modules is repeated execution of instructions, a simple branch speculation is helpful to reduce the overhead due to pipeline flush.

### Mix of MAC and Control code modules
These modules are also computationally intensive but they involve control code operations amidst MAC intensive operations. Transforms operations like Fast Fourier Transform (FFT), Filterbank are typical examples of this category. The processor requirements for this category of modules are:

- **DSP instructions**: Since both the above category of modules, viz. MAC intensive and mix of MAC and control code modules, primarily involve MAC operations, it will be efficient to have MAC and MLS (Multiply and Subtract) instructions preferably without overhead for performing accumulation.

- **Mixed bitwidth operations**: The minimum bitwidth of the operands are determined by various factors. For instance, in filtering opera-
tions the output precision and filter stability determines the bit width of the filter coefficients. So instructions capable of handling mixed bitwidth operands are desirable for an efficient implementation.

**Packed Data processing:** These modules like FFT typically operate using 16-bit coefficients called twiddles. Higher the radix of the FFT, more twiddles are involved but lesser is the relative complexity. These twiddles can be used by packing into 32-bit data to reduce any register crunch in the core loop. Or if possible overhead of repeated loads can be avoided altogether by loading the twiddles onto registers outside the core loop. So instructions capable of operating on packed data are desirable for efficiency.

**Bit Reversal:** Instructions capable of performing bit reversal are beneficial for key modules, for instance FFT, where bit reversed addressing is needed.

**Control code modules**
These modules mainly involve control code and the code flow is data dependent. Bistream demux and entropy decoding operations are examples of this category. For instance, MP3 decoder has Huffman decoding and WMA Decoder has Run length decoding. The processor requirements for this category of modules are discussed next.

**Unsigned and Bitwise operations:** These modules operate on the packed bitstream, which require data extraction. Instructions for performing bitwise operations, extract, packing, unpacking and instructions capable of handling unsigned operands are desirable for an efficient implementation.

**Compact footprint:** The code flow is not predetermined in these modules, since it is mainly data dependent. So having a smaller footprint helps reduce any overhead due to cache flush.

**ARM Cortex-M3 and Cortex-M4 Processors**
In this section, the features of Cortex-M3 core, the low power yet high performance microcontroller from ARM, will be explored and then Cortex-M4, the latest processor core in the Cortex-M series, will be discussed. The features of Cortex-M processors with example instructions and cycles are listed in Table 2.

**Cortex-M3 Features:** The high precision MAC unit with a precision for multiplying up to 32-bit with 32-bit data giving 64-bit accumulated result is present. Other advantageous features like DSP instructions, instruction to assist norm calculation, branch speculation, high code density due to Thumb®-2 and hardware divide instructions are present. These features are advantageous for audio processing.

**Cortex-M4 Features:** The Cortex-M4 core, successor to Cortex-M3, inherits all its features, and has additional ones like single cycle MAC unit and floating point unit (FPU). The MAC unit takes just 1 cycle for even highest precision operations, multiplying 32-bit by 32-bit data giving 64-bit accumulated result. With new features like DSP instructions with SIMD capability, high precision MAC unit, Cortex-M4 is a microcontroller with DSP capabilities, a Digital Signal Controller (DSC). These features are advantageous for audio processing, where high precision is desirable to maintain transparent audio quality.

### Table 2: Cortex-M3 and Cortex-M4 features, example instructions and cycles

<table>
<thead>
<tr>
<th>Feature</th>
<th>Example Instructions</th>
<th>Cortex-M3 cycles</th>
<th>Cortex-M4 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>High precision MAC</td>
<td>SMULL, SMLAL UMAAL</td>
<td>3-7</td>
<td>1</td>
</tr>
<tr>
<td>DSP Instructions</td>
<td>MUL, MLA SMMUL, SMMLSR</td>
<td>1-2</td>
<td>1</td>
</tr>
<tr>
<td>Saturated arithmetic</td>
<td>SSAT, USAT QADD, QSUB</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bitwise operations</td>
<td>SXTH, UBFX PKHTB</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Mixed bit-width capability</td>
<td>SMULWB, SMLWT</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Packed data processing</td>
<td>SMULBB, SMLABT</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>SIMD capability</td>
<td>SMLAD, SMUADx</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

**Code Examples of Audio Modules**
Now, a few code examples of routines from different audio processing components—an audio decoder, encoder, post processing, and general signal processing will be shown. The instructions in the code segment in **bold typeface** are newly introduced in Cortex-M4. The number of cycles taken by each instruction is given in braces. Summary of the total cycles taken, number of instructions required, and registers used are tabulated for each example.

**Window Overlap Add module**
Windowing with Overlap add is a typical module present in most audio decoders. It is computationally intensive and characterized by MAC operations. Figure-4 illustrates the processing of this module. The implementation in this example is given in Equation 1.

\[
V_{16}[i] = \text{Old data} \ast \text{New data} = \text{Frame Output}
\]

[Figure 4: Diagram of Window Overlap Add module](#)
Biquad IIR Filter module
Biquad is the common name given to second order IIR filter, a typical routine present in the audio encoders and post processing modules like equalizer. It is computationally intensive in nature and characterized by MAC operations. Figure-5 illustrates the Direct Form-1 version of this module. The implementation in this example is given in Equation-2. In this example, the register usage and number of instructions needed are same on both the processors.

Figure 5: Biquad Direct Form 1 IIR Filter

Equation 2: Window Overlap Add module

\[ x_{32}[i] = c_{32}[i] \cdot w_{16}[i] + p_{32}[i] \cdot v_{16}[i]; \]
\[ W_{16}, V_{16} = 16 \text{ bit window coefficients}; \quad 0 < i < \text{ frame length} \]
\[ C_{32}, P_{32} = 32 \text{ bit input data from current and previous frame} \]

Radix-2 FFT Butterfly module
FFT is a general signal processing operation. It is typical of audio processing modules which need FFT operation for frequency domain transformation. It is computationally intensive and characterized by mix of MAC and control code operations. This example only illustrates the Radix 2 decimation-in-time (DIT) butterfly. The input and output data are in 16 bit precision. Figure 6 illustrates the processing of this module. The implementation in this example is given by Equation-3.

Figure 6: Radix-2 DIT FFT Butterfly

Equation 3: Radix-2 Decimation-in-Time Butterfly

\[ a = a_0 + W_N \cdot b; \quad b = a_0 + jW_N \cdot b; \quad W_N = w_{r} + jw_{i}; \]
\[ c = c_0 + jC_N \cdot c; \quad A = A_0 + jA_; \quad B = B_0 + jB_; \]
\[ A = a_0 + W_N \cdot b; \quad B = a_0 + jW_N \cdot b; \quad C = w_{r} + jw_{i}; \]
\[ a = a_0 + W_N \cdot b; \quad A = a_0 + jA_; \quad B = a_0 + jB_; \quad C = w_{r} + jw_{i}; \]
**FIR Filter module**
FIR filtering is a typical signal processing operation and usual of post processing modules. This module is computationally intensive and characterized by MAC operations. Figure-7 illustrates the processing of this module. The implementation in this code example is given in Equation-4.

![Diagram of FIR Filter module](image)

**Figure 7: Diagram of FIR Filter module**

\[
y_{32}[n] = \sum_{i=0}^{N} x_{16}[n-i] \cdot c_{16}[i]
\]

\[y_{32} = 32 \text{ bit filter output; } N \text{ - filter order}
\]

\[x_{16} = 16 \text{ bit input data; } c_{16} = 16 \text{ bit filter coeff}
\]

**Equation 4: FIR filter equation**

---

**Audio Solution on ARM Cortex-M Processors**
Ittiam’s audio solution on Cortex-M processors comprises of low-power audio codecs and high quality post processing components. The typical performance for a few chosen set of representative audio components on both Cortex-M3 and Cortex-M4 processors are given in Table 3.

**Processor features for audio:** The audio codecs have been optimized by making the best use of the DSP instructions that Cortex-M3 offers and SIMD capability that Cortex-M4 processor offers. The performance optimized audio solution enables efficient use of processor MHz, resulting in longer audio playback time. The audio post processing components achieve high quality by efficiently utilizing the high precision MAC unit. These features of Cortex-M processors enable Ittiam’s audio solution to achieve high quality at low power.

**Audio Capability:** These typical performance numbers are for stereo mode and 44.1 kHz sampling rate. The decoders and the encoder are configured for 128kbps. The parametric equalizer is configured for 3 bands. All the performance numbers are assuming zero-wait state memory configuration. The performance on hardware will vary with actual memory configuration.

**Table 3: Processor resource utilization for audio components on Cortex-M3 and Cortex-M4**

<table>
<thead>
<tr>
<th>ITTIAI Audio Components</th>
<th>Cortex-M3</th>
<th>Cortex-M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical MHz</td>
<td>% Load</td>
<td>Typical MHz</td>
</tr>
<tr>
<td>MP3 Decode</td>
<td>15 – 18</td>
<td>16%</td>
</tr>
<tr>
<td>Audio Decoders</td>
<td>AAC Decode</td>
<td>21 – 23</td>
</tr>
<tr>
<td>WMA Decode</td>
<td>MP3 Encode</td>
<td>10 – 15</td>
</tr>
<tr>
<td>Audio Encoder</td>
<td>Parametric Equalizer</td>
<td>Stereo Widening</td>
</tr>
</tbody>
</table>

**Table 4: Memory resource requirement for audio components on Cortex-M3 and Cortex-M4**

<table>
<thead>
<tr>
<th>ITTIAI Audio Components</th>
<th>Cortex-M3</th>
<th>Cortex-M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM (kB)</td>
<td>ROM (kB)</td>
<td>RAM (kB)</td>
</tr>
<tr>
<td>MP3 Decode</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>WMA Decode</td>
<td>26</td>
<td>45</td>
</tr>
<tr>
<td>Parametric Equalizer</td>
<td>2</td>
<td>20</td>
</tr>
</tbody>
</table>

**Conclusion**
From the analysis of the various audio processing modules, and the processor requirements for audio processing, we can conclude that the features and capabilities of ARM Cortex-M3 and Cortex-M4 processor cores make them efficient for processing audio. The Cortex-M processors are well suited for audio applications given their low-power and superior performance. These processors are very much audio capable and they can be used for low-power design enabling longer audio play out time while simultaneously providing an enriching listening experience.