The Benefits of Using MIPS® Processors for Consumer Audio Applications

by Rajesh Palani and Radhika Thekkath

MIPS Technologies, Inc.

Consumer devices such as mobile audio players, set-top boxes (STBs), digital TVs (DTVs), and digital versatile disc (DVD) players and recorders are typically implemented using a multi-function system-on-chip (SOC). Such an SOC performs two primary functions: application processing and video/audio signal processing. The application processing (or host processing) is usually handled by a programmable core such as a MIPS® processor. Video signal processing, due to its degree of computational complexity, is done using dedicated hardware. Audio signal processing is somewhat less computationally demanding than video, and in the past, has been handled by hardwired logic or a digital signal processor (DSP).

However, audio subsystem requirements have increased because consumer products need to support more complex algorithms, advanced pre/post processing, and full-duplex encode decode simultaneously. As audio requirements grow, processor frequencies increase due to architectural innovations enabling programmable processors such as those from MIPS Technologies to execute demanding audio applications along with host functions.

It is possible to build two types of SOC architectures to execute audio applications on a MIPS core: (1) one CPU that does host processing and a second CPU dedicated to audio processing (see Fig 1), or (2) a single CPU that does both host and audio processing. There are significant benefits associated with using a MIPS processor for audio processing. Most notably, this type of solution provides:

- **A single processor architecture for highly integrated SOC solutions**
- **A reduction in the total SOC design and manufacturing cost**
- **A programmable audio processor that extends the life of the SOC** design
- **A time-to-market advantage**, and
- **The MIPS™ Soft Audio Interface, which facilitates application software development and integration**

Multi-Core SOC Architecture for Audio

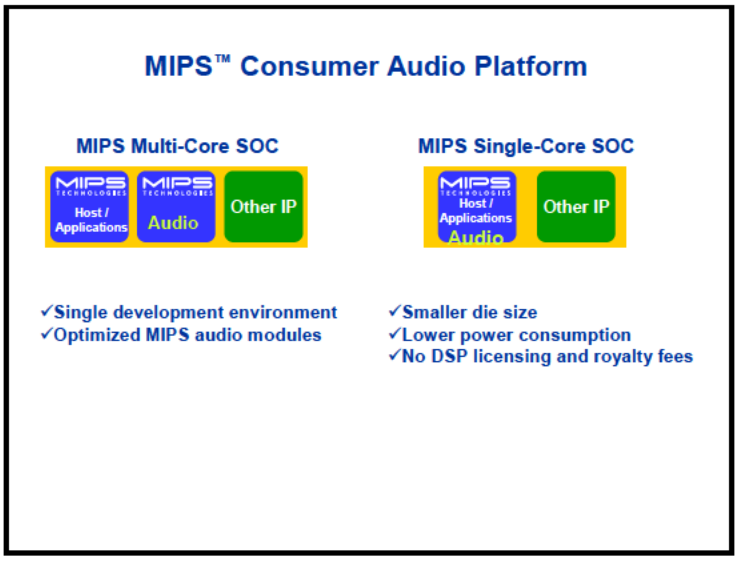
In the multi-core scenario the **host CPU** runs the operating system, end user applications, and services, while **a dedicated audio processor runs** the audio processing function. Several MIPS Technologies licensees have SOC implementations for consumer electronic devices that use a dedicated MIPS core for audio processing.
In general, using a MIPS processor for audio provides a programmable solution. Programmability implies that an existing design can be adapted easily for a variety of audio algorithms. This provides two key benefits: the ability to target a single SOC for multiple end-user applications, as well as extend the life of the SOC design since audio standards continue to evolve and change rapidly.

An advantage of a dedicated audio processor is that it does not compete for CPU cycles with other applications and therefore has a lot of headroom. This headroom on the audio co-processor can be used in multiple ways: (1) the spare CPU cycles can be used to encode and decode multiple streams simultaneously, (2) it can be used to ensure the highest audio quality in high-end designs, or (3) the headroom can be used to reduce the frequency of the processor by reducing the voltage, and thus lower the power consumption of the audio sub-system and the entire SOC.

The advantage of using a programmable processor for audio algorithms is particularly apparent with this configuration, since it provides a scalability that can address constantly evolving worldwide audio standards. With a dedicated DSP, current performance and power needs may be satisfied, but they may not scale to meet future requirements. MIPS processors for audio offer a clear and easy migration path to higher-performance processors, which are binary-compatible with their predecessors.

When audio processing is done on a core separate from the host CPU, a communication mechanism is needed between the host and audio processor. Building a communication interface between two MIPS cores is much simpler than building one between a MIPS CPU and a DSP. For example, the LL/SC (Load-Linked/Store-Conditional) instructions available on MIPS CPUs can be used very easily to build a communication/synchronization mechanism.

Figure 1 Audio Processing Architectures



Single-Core SOC Architecture for Audio

In a **single-core environment**, the greatest benefit is derived by **completely eliminating the DSP or hardwired** audio block and **executing the audio on the MIPS host processor**. This reduces the die size and overall system debug time. This, in turn, ensures reduced cost and quicker time-to-market.

With a real-time operating system (RTOS) running on the host CPU, the audio processing can be done with one of the threads (tasks) of the system. The RTOS must ensure that this audio thread is given sufficient scheduling slots to complete its task in a timely manner. Often, the audio processing will only require a fraction of the CPU cycles; hence, this requirement can be met easily. But this single processor, in addition to the RTOS, can also be executing other applications such as video control. Appropriate scheduling mechanisms are needed to ensure that all tasks complete on time.

In a single-core solution, the performance of the audio application can degrade due to interference in the instruction and/or the data cache. This degradation is a function of the specific combination of operating systems and other control functions executing on the processor. If the performance drop is unacceptable, then this can be addressed using one of two possible methods. Once the cause of degradation is isolated to either the instruction or the data cache, the first method requires locking down appropriate cache lines that hold critical functions or data arrays of the audio application.

If the cache line-locking method is not desirable, then a dedicated Scratchpad RAM (SPRAM) can be used. SPRAM is an implementation choice in MIPS processors that provides predictable low-latency access to an on-chip memory. The size of the SPRAM can be much larger than the cache size if needed, but often a small size SPRAM can have a significant performance benefit to an application. By loading the "text" (code) sections of critical functions of the software audio decoder in SPRAM, the instruction cache

misses in the audio decoder can be reduced. Alternately, the SPRAM can be used to hold commonly used data arrays, which could decrease data cache misses.

Comparison between Single-Core and Multi-Core SOC Architectures for Audio

	Single-Core Audio	**Multi-Core Audio**
Target consumer device or application	Those that are not expected to require a lot of headroom	Those requiring a lot of headroom now or in the future
Target markets	Low-end	Mid- to high-end
Power requirements	Reduces power by eliminating the DSP IP block from the SOC	Reduces power by customizing core frequency to meet audio requirements
SOC die size	Lower chip area by eliminating the DSP	Extra chip area for the dedicated audio core
Inter-processor communication	No extra support required, use a memory location as a synchronization flag	Use the LL/SC MIPS32® architecture instructions to build a synchronization flag

Lowest-Cost Total Solution: Using MIPS Processors for Audio

When making decisions about the architecture of a system, several major costs need to be considered such as: **licensing**, **royalty**, **die size**, **development tools**, and **design-time**.

If a single-core SOC architecture can be used, especially for low-end systems, this eliminates the DSP altogether, lowers the total die size, and eliminates the DSP license fee and royalty. This translates to a reduction in overall cost for SOC manufacturers and OEMs.

Whether SOC vendors choose to run audio on a dedicated core or move audio processing onto the host, software development tools costs are lower since the same tool-chain can be used to develop both host-based and audio applications. A major component of the tools cost is the maintenance associated with the tools. In the case of a home-grown DSP, this can be a significant part of the total cost.

Developers who use a MIPS core for control functions and a DSP for audio processing must learn two different development environments. A typical development environment may include the operating system, compiler, simulator, emulator, debugger, trace tool, probe and profiler. However, when the same core is re-used, there is only one development environment to learn, reducing overall design time and, hence, design cost.

Pre-optimized Audio Software Provides a Time-To-Market Advantage

MIPS Technologies and its audio software partners provide a wide range of optimized audio applications for MIPS32® cores. These include many of the standard audio codecs used in consumer applications such as digital cameras, digital camcorders, STBs, DTVs, and DVD players. These highly optimized algorithms, together with high-performance development tools provided by MIPS Technologies, allow developers to focus on driver application development and integration, rather than optimization of standard audio algorithms.

For customers who desire to tune other audio/DSP algorithms, the MIPS™ DSP Library is available as part of the MIPS™ Software Toolkit. This library implements a variety of signal processing functions that have applicability in speech compression, echo cancellation, noise cancellation, channel equalization, audio processing, etc., and include common functions such as filters and FFT. These functions have been optimized for various MIPS Technologies processor families.

There are many features of MIPS cores that allow efficient execution of DSP-like applications. The optimized audio codecs and the DSP Library take advantage of these features to offer important benefits:

- The use of 32-bit integer data for internal computation in all audio algorithms provides the best possible audio quality.
- The MIPS32 Multiply-Accumulate (MADD) instruction is very effective in coding DSP MAC operations.
- The data prefetch instruction in the MIPS32 architecture is used when appropriate in DSP loops. This allows the prefetching of data into the cache for the next iteration, while executing the instructions of the current iteration. This has a significant savings in the total run time since cache miss latencies are avoided.
- When a small memory footprint is desirable for the most cost-efficient solution, the MIPS16e™ Application Specific Extension (ASE) is used to reduce the program code size. This is simply provided as a compile-time option, which offers a significant reduction in program code size.

In addition to the features above, the software uses other techniques to extract the best possible performance:

- Hand-coding of the computation-intensive critical operations in MIPS32 assembly
- Hand-scheduling on a core-by-core basis to minimize *load-to-use*[1] bottlenecks
- Loop unrolling and software pipelining commonly used for the best code scheduling

To offer maximum flexibility, all the algorithms in the MIPS Consumer Audio Platform are implemented in software. However, SOC designers can implement emerging audio standards using the instruction set of a MIPS core via the CorExtend™ feature. User-defined instructions (UDIs), or CorExtend, allow the addition of new instructions and state for application speed-up. This customization capability provides performance improvements as well as product differentiation for an SOC vendor.

Migration Path

MIPS Technologies has a history of continually improving performance, through increasingly sophisticated core design and through architectural advancements. In the core arena MIPS Technologies has improved clock speed performance with the addition of an 8-stage pipeline to product offerings. In the architecture area, MIPS Technologies has improved IPC performance through the addition of the Release 2 Architecture.

[1] A load instruction's data arrives from the cache/memory system after the instruction needing the data starts executing – the processor stalls until the data is available

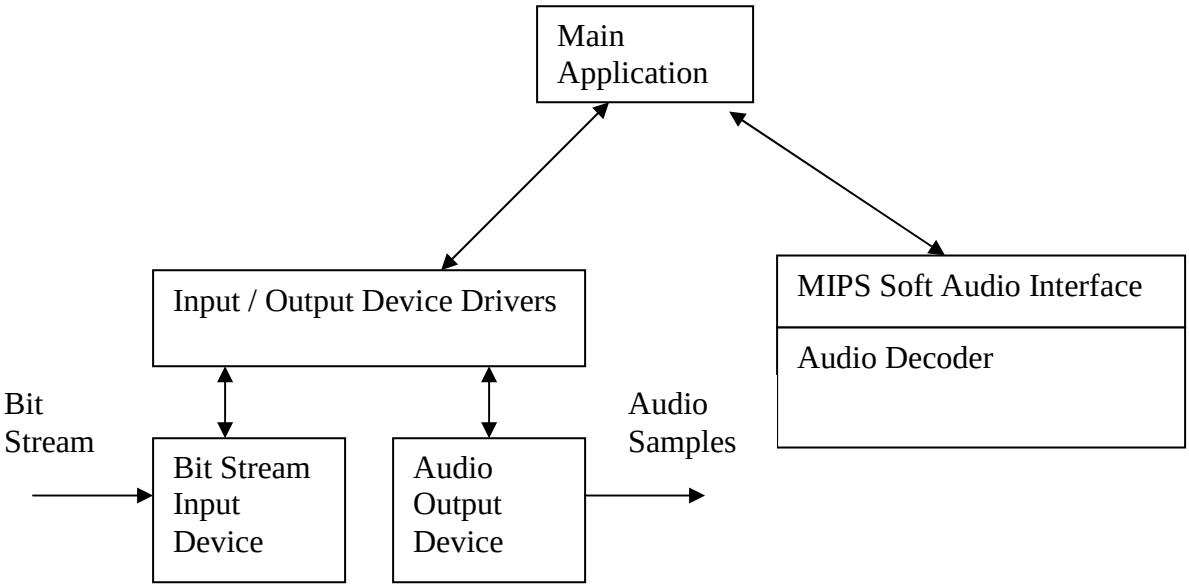
Continuing its commitment to improved performance, MIPS Technologies recently introduced an ASE for Multi-Threading (MT) a MIPS processor. This has particular relevance to the audio marketplace. The MIPS MT ASE improves overall system performance by filling pipeline stalls with useful work from another available thread. But the more significant feature of MT ASE for audio is that it provides mechanisms for quality-of-service scheduling.

The MT ASE has a concept of multiple Virtual Processing Elements (VPEs) that can share a single pipeline. In an audio environment, two VPEs can be used: one to run an OS and the other to run the audio application. The MT ASE allows scheduling policies where specific tasks are guaranteed a minimum allocation of the processor bandwidth. In addition, it provides a Quality of Service (QoS) feature. The combination of these two features can guarantee a real-time execution of the audio application without skipping any audio frames, while also servicing all other real-time tasks as well as the OS in a timely manner.

The QoS feature essentially eliminates the impact of OS interrupts on the performance of the audio function. Normally, interrupt service introduces considerable variability in the execution time of the thread that "takes" the exception. The MT ASE provides a mechanism that causes any asynchronous exception raised to be deferred until an OS thread (non-exempt thread) is scheduled. This increases the interrupt latency in a bounded and controlled manner for OS tasks while preserving the performance of the audio task. If interrupt handler execution takes place only during issue slots not assigned to exempt real-time QoS threads, interrupt service has zero first-order effect on the execution time of such real-time code.

Example Use of a MIPS Audio Decoder

An audio decoder is typically accessed from a driver application. The driver is responsible for extracting the incoming bit-stream from the appropriate input device and for sending the decoded bit-stream to an audio output device, as shown in the figure below. It might also perform other post-processing functions, such as bass management on the decoded audio bit-stream, before sending to the output device. The driver accesses a MIPS audio product via a standard interface mechanism, the **MIPS Soft Audio Interface** (SAI). This common interface allows easy porting of the main application across the various audio decoders from MIPS Technologies.



The MIPS SAI provides the interface functions and structures between the decoder and a main program or RTOS for high-level control and monitoring of decoder operation. The interface provides access to parameters that control decoder operation and access to status and error information. The interface implements three basic functions: one to initialize the decoder, one to read a frame header, and one to actually do the decode processing. For every decoder there is a data structure with pre-determined fields relevant to a specific decoder. The decoder reads and updates the structure for every frame. At the beginning or end of a frame the main program can access the data structure to read status or change control parameters. Refer to the example code below which illustrates the operation of a driver program. The functions in the MIPS SAI are highlighted.

```
int main(void)
{
      decoder_specific_struct dec_ptr;

      //Initialize the decoder.
      mips_sai_dec_init(&dec_ptr);

      //Open the bitstream input device.
      open_input();

      //Open the audio output device.
      open_output();

      //Allocate buffers for decoder operation.
      allocate_buffers(&dec_ptr);

      while(not-end-of-input-bitstream)
```

```
            //Read the input bitstream.
            read_input(&dec_ptr);

            //Read the frame header for encoded stream parameters.
            mips_sai_dec_readheader(&dec_ptr);

            //Set up params for decoder operation.
            setup_params(&dec_ptr);

            //Decode the frame
            mips_sai_dec_process(&dec_ptr);

            //write the decoder output to the device driver.
            write_output(&dec_ptr);

      end-while

      //Close the input and output devices.
      close_input();
      close_output();

      return 0;
}
```

Summary

The various features available on MIPS processors offer a number of benefits when executing consumer audio algorithms. These include a single programmable architecture, lower system cost and reduced time-to-market. This makes MIPS processors a viable and attractive solution for audio applications in consumer devices ranging from battery powered mobile audio players to high performance DVD recordable devices.

# # #

For more information, please visit the company's website at www.mips.com.