Using audio codecs IP as the digital audio hub in mobile multimedia systems

Carlos Azeredo Leme, Manuel Mota - April 23, 2013

Introduction
Mobile multimedia devices process and combine audio signals from a variety of sources, including the baseband processor, Bluetooth enabled devices, and WiFi networks. The result is that today's smartphones and tablets are 'digital audio hubs' that must receive multiple asynchronous digital audio signals, synchronize them, and output them on loudspeakers or headsets.

Tablet/smartphone system-on-chips (SoCs) designers are faced with the challenge of implementing the complex audio mixing functions in the most cost-effective way possible. Traditionally, this function can be implemented in the application processor or in a dedicated audio processor; however, this is not the most cost-effective way to use the limited processing resources available to the system.

By integrating an audio analog codec that implements the 'audio hub' functionality and is able to process and mix audio signals from asynchronous sources, system designers can free the scarce main processor resources for more relevant tasks and simplify the system design, thus achieving a more effective solution.

This article will analyze:

- The benefits of having the audio codec in mobile multimedia systems operating as a digital audio hub to interconnect the different audio signal sources and destinations, each having independent clock domains
- How to synchronize and combine the various audio streams originated by different sources in the system, using built-in asynchronous sample rate converters (ASRCs).

By leveraging the latest improvements in audio codec IP, designers and system architects will be able to deliver the tightly integrated solutions that will make their SoCs stand out from the competition while minimizing costs.

Audio Codec Requirements
The core of an audio codec is composed of two types of data converters: an analog-to-digital converter (ADC) for recording and a digital-to-analog converter (DAC) for playback. For a stereo or multi-channel codec, these are replicated accordingly. Figure 1 shows a typical block diagram of a stereo audio codec.
In the analog side, the record channel includes amplifiers with volume controls to bring both the weak microphone levels and the large interconnect line levels to the input range of the ADC. The playback channel includes output drivers able to directly connect to earphones or to small speakers, each with its respective volume controls. There is also a low-noise power supply for microphone biasing.

The digital side includes multiple blocks. The most important are the digital audio filters that convert the data rate to the oversampled clocks of the data converters and remove the high-frequency noise outside the audio band. Also important is a clock management block, which ensures that all multi-rate blocks are synchronized with each other and support the required sampling rate combinations.

**Audio Codec Operating as a Digital Audio Hub for Mobile Multimedia**

Audio Codec Operating as a Digital Audio Hub for Mobile Multimedia

In a modern mobile multimedia system, there are likely to be a number of digital hosts, all operating on their own clock domains. For example, the baseband processor will manage communications with the cellular radio, the applications processor will manage media files from the system's flash memory, and the Bluetooth integrated circuit (IC) will connect to any number of Bluetooth-enabled peripherals. Each of these digital hosts is operating on separate, and asynchronous, clock rates.

However, any of these audio streams may need to be mixed and transmitted to the analog output or to another host. For example, the user may be listening to a music file coming from the application processor on a headset. The music file must be mixed with signals coming out of the communications baseband indicating an incoming call.

Mixing separate audio streams can be done through signal processing in dedicated processors, or directly in the applications processor, at the expense of adding hardware or an additional burden on the embedded applications processor.

It is possible to eliminate this burden and simplify the system by moving these functions to the audio codec. An audio codec in such a system is not only bridging the digital domain to the analog domain,
but it is also operating as a digital audio hub that seamlessly interconnects different clock domains and is able to mix audio streams from each of them.

An example of a digital hub is shown in Figure 2, which illustrates an audio codec that is connected to three different audio hosts, each synchronous to its own external clock domain.

The audio codec uses internal synchronizing blocks, or ASRCs, to convert the data rate from each audio host to the internal clock domain. All mixing is carried out in the internal clock domain. The resulting audio stream can also be mixed with digitized content from a microphone and finally converted back to analog for playback on speakers or a headset, or alternatively be sent out to one of the audio hosts, after resynchronization with the ASRC.

![Audio Clock domain](image)

**Figure 2: The Digital Audio Hub - Digital-centric audio processing with multiple digital audio hosts**

Furthermore, the audio codec operates as a digital audio hub between the different clock domains and can select the lowest jitter clock available in order to achieve the best audio performance.

In modern audio codec implementations suited for SoC integration at 28 nanometer (nm) and below, all the re-synchronization, mixing, and filtering functionality is carried out in the digital domain. The conversion to analog is done only when connecting to the external speakers or microphones.

By taking full advantage of the area and cost scaling enabled by digital circuitry in 28 nm and below, these digital-centric audio codec/hub architectures are ideally suited for integration in the most advanced mobile multimedia SoCs.

**Using ASRCs to create Digital Audio Hubs**

Digital audio is sampled at a multitude of standard rates, depending on the application. The clocks required by the data converters on an audio codec depend on the audio data sampling rates as well
as on the clocks available on the host application and SoC. As shown in Table 1, the combinations are quite complex due to the multitude of audio sample rate options and available host clocks.

<table>
<thead>
<tr>
<th>System</th>
<th>Standard audio sampling rates (Fs)</th>
<th>Oversampling frequencies of data converters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>64 x Fs</td>
</tr>
<tr>
<td>Telephone</td>
<td>8 kS/s</td>
<td>0.512 MHz</td>
</tr>
<tr>
<td>Compact disc</td>
<td>44.1 kS/s</td>
<td>2.8224 MHz</td>
</tr>
<tr>
<td>Professional audio (DAT)</td>
<td>48 kS/s</td>
<td>3.072 MHz</td>
</tr>
<tr>
<td>DVD (6 channel audio channel)</td>
<td>96 kS/s</td>
<td>6.144 MHz</td>
</tr>
<tr>
<td>A-DVD, Blu-ray disc</td>
<td>192 kS/s</td>
<td>12.288 MHz</td>
</tr>
</tbody>
</table>

Table 1: Common over-sampling frequency for standard audio systems

Since the data converters must operate at oversampled frequencies, typically 128X or 256X, the required master clock frequencies to drive the data converters would be in the range of 5 to 12 MHz. An audio codec must, therefore, support a wide variety of audio sample rates and accommodate a range of master clock frequencies. It is not a straightforward objective due to the multitude of combinations and restrictions in the possible clock frequency ratios.

To accommodate all these combinations of data rates and available clocks, different techniques can be used such as generating the audio codec clock from a PLL, or re-using an existing clock, such as from one of the audio hosts and accepting an approximated sample rate.

However, by implementing an ASRC filter in the interface between the different clock domains, the audio codec/hub can accommodate all the common sampling rates without compromising quality.

**Using ASRCs (cont.)**

An ASRC offers flexibility and simplicity when interconnecting different audio clock domains. It is basically a digital filter that alters sampling frequency of the data it is processing. It can change the sampling frequency up (up-sampling) or down (down-sampling) asynchronously.

In an ASRC, one or both of the rates can vary independently and the filter will be reconfigured automatically. It uses a rate estimator to constantly track the ratio between input and output sample rates. ASRCs are commonly used in systems where the receiver is required to "lock on" to a source clocked by a separate crystal oscillator.

The main benefits of an ASRC are:

1. Allows full flexibility when multiplexing between audio sources with different sampling rates. For example:
   - Data at 44.1 kS/s can be multiplexed with other data at 48 kS/s by passing it by an ASRC that is re-sampling to a common clock. Data at the same sampling rate but from asynchronous clocks can also be multiplexed.
   - The incoming audio stream can be resampled on the host clock, which is not necessarily a standard audio frequency. It can be a clock derived from the communications radio on a cell phone, or the USB clock available in most electronic devices.
2. Virtually eliminates the effect of jitter on the audio data when the audio codec clock is of very low
jitter. The ASRC basically computes the average frequency ratio between the incoming and the output clocks and transfers the data accordingly. Only slow varying jitter, below the ASRC cut-off frequency (as low as 1 Hz), gets through to the output data. For example, jittery data coming from an audio optical cable where the clock is embedded in the data can be cleaned to the low jitter of the audio codec clock by passing it through the ASRC.

3. No significant area impact is tied to the implementation of the ASRC. An ASRC is a relatively complex type of filter that is implemented using a relatively large number of digital gates. Basically it is equivalent to interpolating the input data by a very large factor (of the order of 1,000,000X) and re-sampling at the output audio rate. It also uses a form of digital PLL to track the ratio of input to output clock frequencies. This results in a relatively large number of gates operating at high clock rates. Fortunately, the added complexity of adding ASRCs to audio codecs is largely offset by the process scaling, especially at 28 nm and below. In fact, the size of an audio codec in 28 nm is dominated by the analog part, namely the data converters, the analog volume controls and output drivers. Therefore, there is no significant impact in area by including the asynchronous interfaces, while all its advantages regarding flexibility to interface different clock domains greatly simplifies the system integration, especially in large SoCs for smart-phones and ultra-mobile computers.

Summary
Today's smartphones and tablets are 'digital audio hubs' that must receive multiple asynchronous digital audio signals, synchronize them, and output them on loudspeakers or headsets.

Mixing separate audio streams can be done through signal processing in dedicated processors, or directly in the applications processor, adding the expense of extra hardware or a higher burden on the embedded applications processor. It is possible to eliminate this burden and simplify the system by moving these functions to the audio codec. An audio codec in such a system is not only bridging the digital domain to the analog domain, but it is also operating as a digital audio hub that seamlessly interconnects and mixes audio streams from different clock domains.

Synchronizing and combining the various audio streams originating from different sources in the system is handled in the audio codec by implementing an ASRC filter in the interface between the different clock domains, without compromising quality. This has the added advantage of virtually eliminating the effect of jitter by using a low jitter clock on the audio codec, even if the incoming audio streams have high jitter on their clocks. The Synopsys DesignWare® Audio Analog Codec IP implement the audio hub functionality, simplifying the integration of complex audio codecs and helping design engineers build competitive mobile applications products and get them to market faster.

More about Carlos Azeredo-Leme
More about Manuel Mota