An Optimised Software Solution for an ARM Powered™ MP3 Decoder

By Barney Wragg and Paul Carpenter

Abstract

The market predictions for MP3-based appliances are extremely positive. The ability to maintain impressive sound quality whilst reducing the data requirements by a factor of 1:10 or more, has led to an explosion of content on the Internet. Traditionally, a DSP processor may have been specified as an implementation platform for MP3. However, analysis of the key technical requirements for MP3 show that a programmable software solution mapped to a low-power ARM® processor has many advantages over a DSP-based implementation.

Introduction

The ability to compress sampled data and still maintain excellent sound quality is transforming the digital audio market. A new generation of solid-state players, where data is stored in non-volatile flash memory, has rejuvenated the market for portable audio equipment. Generally, these players are more robust than CD products, and have reduced power requirements because they contain no moving parts. They can also be manufactured in a smaller form factor and are better suited for portable personal audio and automotive applications.

Compression techniques can also be applied to the storage of audio data on CD and hard disks. The compressed audio format applied to any media allows considerably more content to be stored than before. For example, it is possible to store around eight and a half thousand minutes of music, the equivalent of 150 uncompressed CDs, using a laptop PC 4.2Gbyte hard disk.

As well as the considerable benefits that MP3 technology can bring to the design of the audio player, the combination of excellent sound quality with good compression means that relatively low bit rates are required to transmit audio content electronically. This has been a major factor in the pervasiveness of MP3 content on the Web, and will undoubtedly continue to be a significant influence in the overall growth of the MP3 market. Estimates vary, but the major manufacturers predict that the market for portable players alone will grow to tens of millions of units by 2002, with a significant number of sales replacing the uncompressed CD format over the next few years.

New channel possibilities for the delivery of MP3 audio content – for example, wireless distribution via 3G or Bluetooth, is likely to drive MP3 functionality into other appliances such as mobile phones, communicators and PDAs.

The ARM MP3 Solution

The ARM MP3 software decoder is fully compliant with the ISO MPEG audio standards. This includes ISO/IEC 11172-3 (MPEG-1), ISO/IEC 13818-3 (MPEG-2), and also the MPEG-2.5 low bit rate extensions (as defined by the Fraunhöfer Institute). The implementation supports both stereo and mono decoding.

The target processors include the ARM7™ and ARM9™ platforms. These are based on the ARM V4T architecture, and offer different performance points. It is important to note that this

implementation does not require the use of a DSP or in fact any other external hardware – the MP3 processing can be done entirely on the ARM.

Adaptability

A programmable solution is particularly suitable for a digital audio platform since it provides greater flexibility, appropriate for a rapidly changing market. This is beneficial where standards are still emerging, for example, the more recent audio codec standards such as MPEG-AAC and WMA, and in the area of digital watermarking for copyright protection. The ability to quickly incorporate modified algorithms can give a manufacturer a significant competitive advantage in bringing derivative products to market.

Low Power and Cost

Since the highest volume MP3 producers will target price-sensitive consumer markets, it is essential that hardware and software cost be minimised. At the same time, MP3 manufacturers must deliver products that meet key consumer purchasing criteria, including excellent audio quality and long battery life.

Delivering market-leading performance in low-power consumption has been critical to the success of ARM in other highly competitive portable markets, such as digital cellular handsets. This core strength of the ARM architecture will scale further with the availability of new silicon process technologies as supply voltages also reduce.

One of the most important benefits of the ARM MP3 solution, and a significant advantage over DSP-based implementations, is that all the required processing can be performed on the ARM as a standalone processor. This helps reduce power consumption, minimises chip area and considerably simplifies the hardware and software development process.

The ARM can perform the audio processing whilst also fulfilling the requirements of the system control functions, such as management of IO, card memory, display and keyboard. In contrast, a DSP-based implementation would require a separate microcontroller to run the rest of the system. Clearly, an implementation based around two processors will require additional chip area. Development of protocols for control and data exchange between the DSP and the microcontroller will increase the overall system complexity.

Integrating all of the functionality onto a single processor is also a critical factor in easing the development process and consequently the time to market.

Ease of Development

Because the ARM processor solution is centred on a single memory system, the availability of a unified memory map considerably simplifies the overall software design task. For systems running an RTOS, it is a straightforward task to call the MP3 function through an API. In contrast, most RTOS do not provide API support for DSPs, and so the DSP-based solution would require development of bespoke scheduling routines – something that is complex and prone to timing difficulties when the task has to be scheduled out to a second processor.

The ARM Developer Suite™ (ADS) provides the software developer with a complete kit of tools including a GUI, C and C++ compilers and linkers in an integrated environment, which can be used to develop application code for the ARM, and also assist with porting RTOS if required. This provides a single, familiar environment for the software designer. Software development for a DSP plus microcontroller implementation may require the designer to use two disparate tool flows. In addition, hardware developers will face a further, complex integration step.

Algorithm Analysis

The software decoder implementation was developed using the ISO reference code as a standard. Although this provides a golden reference for verification of any implementation, it was clear that compiling the standard ISO reference code would yield a far from optimal solution. Analysis of each of the steps involved in decoding the MP3 data was required in order to improve the MP3 implementation for the ARM platform.

The key to the efficiency of MP3 is perceptual coding – in other words ignoring those noises that are masked by more dominant sounds, and therefore not perceived by the ear. In order to do this the MP3 algorithm includes a psychoacoustic model of the human ear. By ignoring information not perceived by the ear, the signal can be represented by fewer bits without loss of quality.

Fixed-point Implementation

Whilst the ISO reference code is based on floating-point arithmetic, for an efficient mapping to most embedded hardware structures it is necessary to consider a fixed-point implementation. The quality of the audio must not be compromised for the product to meet the basic requirements of the consumer. A good system design will maximise the signal-to-noise ratio in the implementation of the algorithm whilst minimising the overall complexity of the design.

As most digital audio players use 16-bit audio DACs, maintaining the audio quality to at least 16-bits is important, otherwise the decoding process starts to introduce noise, which can substantially impair sound quality.

Most consumer DSP approaches use either a 16-bit or 24-bit data path. Using a16-bit DSP is not ideal, as the 16-bit data path has the same precision as the desired output. Consequently, every arithmetic operation during the decode process can add extra noise to the output. A single bit error in the output corresponds to a decrease of approximately 6dB in the signal to noise ratio. Even with extended multiply and accumulate structures, the short 16-bit width of the data path means that accuracy is inevitably lost due to rounding errors, and the lack of resolution results in the introduction of higher levels of noise. In order to avoid this limitation with the 16-bit DSP architecture, the majority of DSP vendors have promoted a 24-bit DSP architecture. Although this does provide more resolution in the data path and internal registers, the disadvantage of this approach is that these devices tend to be more expensive and have higher power consumption than the 32-bit ARM processors.

The ARM has a full 32-bit internal data path and register set, including a 32x32 multiply giving a 64-bit result, which gives it a significant advantage over 24-bit DSP implementations. This means that far greater resolution can be maintained during the intermediate processing

stages, resulting in a highly efficient implementation with the best possible audio quality. Using a 20-bit output the signal to noise ratio of the decoder is in the region of 120dB, which is comparable to CD quality.

MP3 Processing Stages

Analysis of the proposed ISO algorithms at each stage of the MP3 processing led the ARM team to design more efficient alternatives in several cases. This approach enabled significant performance enhancements to be made by modifying the core algorithms to achieve a better mapping on to the ARM instruction set. The processing requirements could be further reduced where common operations could be identified and merged into single steps.

The basic steps in the MP3 decoding process are listed below, and illustrated in the functional block diagram of Figure 1.

1. Acquire header and synchronise
2. Decode header information
3. Unpack scalefactors
4. Huffman decoding
5. Inverse quantisation
6. Inverse transform
7. Polyphase filtering

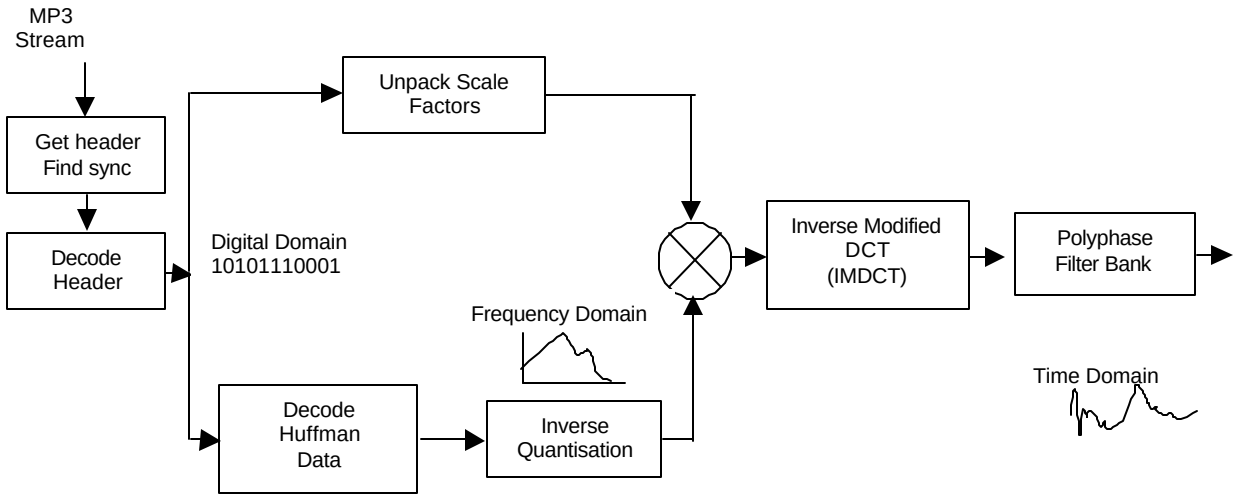

Figure 1. MP3 Processing Stages

Referring to the main MP3 processing stages illustrated in Figure 1, a common assumption is that a special-purpose DSP processor is necessary for implementation. There are several functions – such as IMDCT and filtering, which would commonly be classified as typical digital signal processes.

In fact, on closer inspection many of the operations either involve bit-level manipulation or can be reformulated to better suit bit-level manipulation. In addition, several of the processing steps require execution of complex decision making – for example, cascaded 'if...then' statements. The ARM architecture and instruction set is extremely efficient at conditional execution (i.e. branching on the same cycle as the condition tested), and bit-level signal manipulation.

Several of these efficiencies can be illustrated by considering the execution of the first three steps in the decoding process. Processing starts with identification of the frame sync within the 32-bit frame header, and decoding the information within the header, which typically specifies the MPEG version, layer description and bit rate.

1. Acquire header and synchronise

2. Decode header information

In the ARM implementation these steps are merged into one operation, since the bit-level processing requirements are very similar.

3. Unpack scalefactors

After the header information, the next transmitted data is the scalefactors that control the gain in each frequency band. Unpacking these requires similar processing to steps 1 and 2, and is again based on bit-level manipulation.

One source of efficiency in implementing these steps is the ability to perform 'shift' and 'mask' operations in a single cycle on the ARM's 32-bit barrel shifter. For example, decoding 18 equally sized scalefactors can be done in consecutive single instructions on the ARM:

```
AND   scalefactor, mask, bitstream, LSR #<n>
```

This instruction shifts the bitstream right by the required number of bits and 'ANDs' with a mask in a single cycle. The mask would be a value such as 0x07, which would extract 3 bits. The mask is held in a register, but the same value is used 18 times.

Other architectures usually require two instructions to perform the equivalent operation:

```
LSR   tmp, bitstream, LSR #<n>
AND   scalefactor, tmp, mask
```

The actual frequency energies follow the scalefactors in the bitstream, quantised and Huffman-encoded. The decoder's task is to Huffman-decode, requantise, and transform the energies into the time-domain.

4. Huffman decoding

Huffman coding creates variable length codes, with higher probability symbols assigned shorter codes. Each code has a unique prefix, which means that they can be decoded correctly using a binary tree.

The Huffman encoding in the encoder uses many different Huffman trees selectively according to the data contents, to minimise the total bit-length. The Huffman decoder has to traverse the appropriate tree for each symbol in the frame data to arrive at the decoded value. When the Huffman decoder has decoded the values, they have to be re-scaled using the scalefactors into real spectral energy values.

The ARM implementation uses a proprietary coding of a level-compressed Huffman tree. Implementation of the Huffman decode offers a design trade off between the size of the look-up table and the access overhead. This implementation optimises the size of the ROM look-up in preference to the access time, since this is a relatively fast operation on the ARM.

The in-line barrel shift is capable of a load and offset operation in a single cycle, which facilitates efficient implementation of the look-up operation from the extracted bits within the bitstream.

5. Inverse quantisation

Quantisation is done via a power-law quantiser. In this way, larger values are automatically coded with less accuracy, and some noise shaping is already built into the quantisation process. The inverse quantisation algorithm requires that the value from the Huffman decode is raised to the power of 4/3. The range of input values varies from 0 to 8191, which would require 32kbytes of look-up table storage. However, by manipulating larger values prior to applying the power-law re-quantisation, it is possible to reduce the table size to just 4kbytes. Essentially, the input value is tested – if it is greater than 1023, it is divided by 8 before look-up, and the result is multiplied by 16 to yield a value equivalent to $x^{4/3}$. Because the value from the Huffman decoding operation is often ±1, testing for this condition is worthwhile as in this case no scaling operation is required.

As the most efficient solution again involves the use of a table look-up technique, it was possible to merge the execution of this step into the Huffman decode operation.

Much of this arithmetic can be implemented very efficiently using the barrel shifter. In addition, these operations require complex conditional execution in the control flow of the processor –

e.g. if input > 1023 then ‘divide by 8’
if input = ±1 then ‘don’t scale’

Because the ARM RISC architecture can perform conditional execution – i.e. branching on the same cycle as the conditional test, this further enhances the processing efficiency for this and other similar stages where the control-flow overhead could otherwise be significant.

6. Inverse Transform

To synthesise the output samples, a transform is applied that is the reverse of the time-to-frequency transform used in the encoder. The core of the inverse modified discrete cosine transform (IMDCT) calculation is performed by multiplying an input vector of size *n* by a matrix of size *n* by *n*. Analysis of this operation yields a calculation of complexity:

n additions + n^2 multiplications + *(n-1)n* additions.

Again, it is possible to re-formulate the operation to better suit the ARM instruction set. The ARM proprietary implementation reduced the cost of this operation to complexity $n.\log(n)$ multiplies. For large values of n, this equates to less than 10% of the above complexity definition.

Processing improvements can also be achieved by considering the requirements of the downstream filtering steps. It became apparent that re-ordering the output from the IMDCT simplified the implementation of the polyphase filter.

7. Polyphase Filter

After performing the transform in the decoder, the final operation required before output is the polyphase filter bank, which reconstructs the audio signal from 32 equally-spaced frequency bands. Each band is band-pass filtered using a common low-pass filter modulated by a cosine term.

The ISO standard specifies that the last 2048 samples should be stored for each channel. Re-ordering the data from the IMDCT reduced that requirement to 1024 samples per channel, with no loss in accuracy. This obviously halves the RAM storage required for this operation.

The target architecture is capable of implementing a memory load in 3 cycles and store in 2 cycles. However, by organising the data so that memory access is via consecutive addresses it is possible to achieve effective load and store operations in a single-cycle, which significantly enhances the performance of the convolution operation.

Performance

Table 1 illustrates for three of the popular ARM RISC architectures the required clock speeds to meet the given sample rate. These performance figures assume that the memory is 32-bit wide, zero wait-state memory. The memory required for each implementation is low – just 27kB of ROM and 21kB of RAM, which includes all tables, a 4Kbyte PCM output buffer and a single frame input buffer.

Sample Rate	Content	Channels	ARM7TDMI™	ARM9TDMI™	ARM9E™
48kHz	320kbits (peak)	Stereo	29MHz	25MHz	19MHz
44.1kHz	128kbits (average)	Stereo	25MHz	22MHz	18MHz

Table 1. Peak and average performance figures for MP3 decode on ARM platforms

The peak performance figures are quoted for worst-case Huffman codes whereas typical music is less difficult to decode than a worst-case bitstream.

The ARM9E architecture has enhanced DSP features including a faster multiplier, and can therefore implement the IMDCT and filtering functions in fewer clock cycles than the other architectures. The ARM9E combines the best features of the RISC architecture, with efficient DSP operation.

Because the time spent implementing these particular functions is approximately proportional to the sampling frequency, reducing the sample rate has a less pronounced effect on the overall system clock for the DSP-enhanced (ARM9E) architecture.

Quality

The software implementation has been rigorously tested against the ISO reference standard. The test strategy included use of ISO test patterns and in-house tests, with the aim of stressing all possible corner cases, for all stages of the decoder, such as the Huffman decode function. In order to ensure that all table entries were tested at least once, around 150 different test cases were devised, each comprising of between one and approximately 1000 frames of data. The code was also stressed by deliberately introducing errors into the MP3 bit stream before replaying the data, as well as attempting to replay invalid and non-MP3 files, to ensure there were no adverse consequences such as assertion failures, infinite loops or out of range memory accesses. In addition, the tests verified the quality of the implementation at a range of input bit rates and output sample rates.

The ARM implementation has a maximum difference of +/-1 in the least-significant bit, compared with the reference. Consequently the signal-to-noise ratio of the ARM decoder is approximately 96dB. This represents the best achievable signal-to-noise ratio for a truncated 16-bit output.

Integration

The ARM MP3 software implementation is easy to integrate, with or without an RTOS. A simple API is provided with four functions:

- initialise
- find sync word
- decode frame header
- decode frame data

The development kit also includes an example of a player application demonstrating implementation of forward and backward cueing functions.

The MP3 ARM implementation is commercially available today. Implementations are also available for the current WMA and MPEG AAC standards.

Summary

Examining the implementation of the core MP3 processing has highlighted several features in the ARM instruction set which are critical in achieving a highly efficient solution for MP3 decoding.

To summarise, these are:

- Conditional execution
  - Efficiencies gained in eliminating branching illustrate the significant amount of control overhead within the overall MP3 processing stages.
- Single-cycle bit manipulation instructions
  - Since many of the data processing requirements can be answered by bit-level manipulation, this gives the ARM architecture a significant advantage.
- In-line barrel shifter
  - The barrel shifter is used heavily in the front-end bit stream processing for decoding and scaling operations.
- 32-bit datapath
  - The 32-bit data representation ensures that the final accuracy easily meets the ISO reference.
- Efficient multiple data handling
  - The high bandwidth 32-bit bus can be used to load two 16-bit words in a single cycle, the efficient bit-manipulation functions ensure that the words can be easily extracted.
- 8-bit Booth's multiplication
  - An estimated 25% of the total algorithm relies on parallel multiplication that cannot be efficiently implemented by bit manipulation. Although the number of multiplications required is significant, they do not represent the most critical part of the algorithm. The standard RISC architecture includes an efficient Booth's multiplier. For applications that are more dependent on fast multiplication, the enhanced architecture includes a load-multiply instruction that effectively saves 3 cycles per multiply.

Analysis of the processing stages in the MP3 algorithm illustrates that in the critical front-end steps, which include reading of the bitstream, Huffman decoding and inverse quantisation, the ARM RISC architecture has a performance advantage over many DSP implementations.

Providing a single-processor integrated solution offers power, cost and development advantages over a two-processor implementation based on a DSP interfaced to a microcontroller.

The ARM platform has already established itself in a range of high-volume consumer applications that require a combination of flexibility, low-power and cost single-chip applications. The ARM-based MP3 solution presented here satisfies all of these criteria, and also offers an accelerated development path critical in meeting current market demand.

