

A Full Custom CMOS IEC-986 Audio Decoder with Digital Volume Control

Timothy ADAMS, Richard MASSEY, Divya RAMACHANDRAN, and Richard WINGFIELD

Abstract—This paper describes the design of an IEC-986 audio decoder, and its CMOS implementation. The chip is designed to decode serial audio data in the IEC-986 audio format. Our chip decodes the input stream and produces two 21-bit parallel digital output signals, which can be connected to digital-to-analog converters. The design also includes a 5-bit variable gain feature, which can be used to select 32 different gain levels.

I. INTRODUCTION

We have created a decoder that takes a serial IEC-986 data stream and produces two parallel 21-bit digital audio signals. This output can be easily converted to analog format with two digital-to-audio converters. Our chip also provides real-time digital volume control with thirty-two different linear gain settings. The IEC-986 format is first discussed and then we talk about our chip design and how our chip can be used to produce sound from an IEC-986 or consumer type AES/EUB input stream.

A. IEC-986 Audio Interface

The IEC-986 interface is a way that digital audio data can be transmitted serially through a single transmission line. Data is transmitted in channel status blocks, frames, and sub-frames. Because all data is transmitted in biphase-mark encoding, the information is independent of polarity. Biphase-mark encoding is a different way to transmit ones and zeroes across a serial transmission line. To transmit a '1' in this format, there is a transition in the middle of the data bit boundary. If there is no transition in the middle, the data is considered a '0'. See Figure 1.

Another important rule in biphase-mark encoding is that the signal switches polarity at every data bit boundary. This can be seen most easily in Figure 1 where the three zeroes are being transmitted. A sub-frame is the basic unit into which digital audio data is organized. It contains four preamble bits, four auxiliary data bits, 20 audio data bits, and four other bits that are used for error checking, validity, and control. This makes each

sub-frame contain a total of 32 bits. An example of a sub-frame can be seen in Figure 2.

The preamble can be found by checking for biphase-mark encoding violations. Once detected, this signal can be used to determine which of the two audio channels receives the sub-frame. In consumer-type audio compact discs, only 16 of the 20 audio bits are used. The validity bit signifies whether an audio signal is fit to be converted to analog. This bit is active low. The parity bit generates even parity. The user data bit and channel status bit are accumulated with every sample for each channel. The user data bit is undefined and can be used for any purpose. The block of information obtained from the channel status data gives information about the audio data and its transmission link. Both the user data block and the channel status data block are 192 bits long.

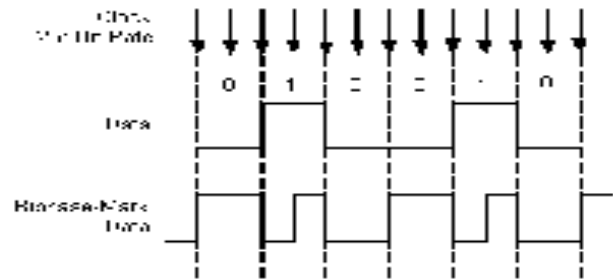


Fig. 1. Biphase-Mark Encoding

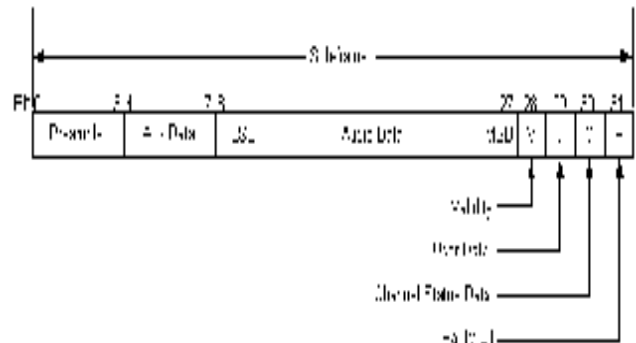


Fig. 2. Sub-frame Format

Two consecutive sub-frames make up a frame. A frame contains both left and right channel data. A block is made up of 192 frames. This is due to the channel status data block and the user data block. See Figure 3.

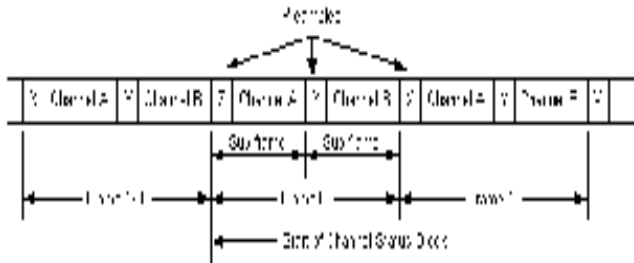


Fig. 3. Frame/Block Format

Just as each channel has a preamble that contains two biphasic-mark violations, the beginning of each block has its own preamble. This preamble is also seen as left channel data. Each of the preamble waveforms can be seen in Figure 4.

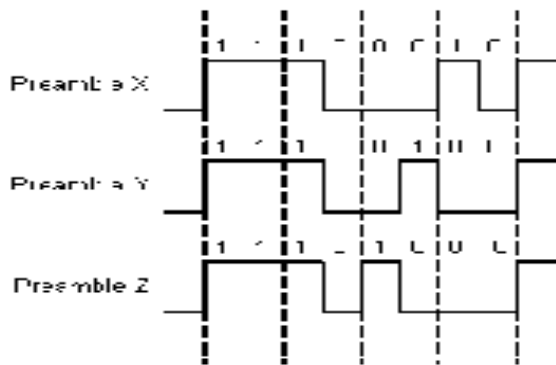


Fig. 4. Preamble Waveforms

B. Functionality

Using this information, we designed a multi-functional chip. This chip receives as input the aforementioned IEC-986 data stream and 5 volume data bits. Our chip amplifies the audio data signal according to the volume bits, but can also be used as a standalone biphasic-mark decoder. Each audio signal channel is given to a digital-to-analog converter (DAC) and can then be output to a speaker, or can be further processed by other digital circuitry.

Our chip is capable of decoding IEC-986 audio data, but ignores the extra information encoded in the auxiliary data field, the user data bit, and the channel status bit. These fields do not contain any

information necessary to produce audio data. It also ignores the parity bit, as it does not perform error checking.

C. Target Applications

IEC-986 encoding has long been established as the standard for consumer audio compact disc players. Because our chip only decodes the first 16-bits of audio data for each channel it may not be suited for some professional audio decoding if that data is encoded with more than 16-bits of audio information per sample. However commercial compact disks only carry 16-bits of audio data per sample for each channel, so our chip is well suited to be used in standard commercial compact disc players. Because these devices are intended to be mass-produced, cost is essential. Our device can lower manufacturing costs by providing the IEC-986 decoding as well as gain control on a single small and inexpensive chip. Since our chip also provides this digital gain control it is suited for audio devices where the volume is to be controlled digitally. These devices may range from portable disc players, to car stereos, or more complex sound systems used in private homes or studios.

II. FEATURES

Our system has a number of features that make our chip useful and easy to interface with an IEC-986 stream. Perhaps the most useful feature of our chip is the 5-bit parallel digital gain control. Our chip also has the ability to automatically synchronize with each frame of audio data. So even if there are errors in the IEC-986 data stream, our chip will automatically align itself with the first valid data frame. Our chip detects the validity bit, and if set high, does not update the audio output. Another useful feature of our chip is that the biphasic-mark clock drives the system clock, so our circuit can play audio data with a variable sample rate. Finally, we route the decoded biphasic-mark stream to an output pin so our circuit can function as a biphasic-mark decoder, making our chip multifunctional.

A. Digital Gain

Since portable electronics are intended to sell at large quantities and at lower costs, incorporating sub-assemblies into a single integrated circuit oftentimes can reduce device cost and size. Another advantage of having a digital gain control unit

instead of an analog one is that it eliminates distortion, which can be caused by traditional analog amplifiers.

The gain control is implemented with an array multiplier. Serial data is input in 64-bit frames, which contain a sub-frame for the left channel and a sub-frame for the right channel. Each sub-frame contains 16-bits of audio data. The gain control is 5 bits wide, allowing for 32 different gain levels. These levels vary linearly. After each sub-frame is read in, the 16-bit audio data is multiplied by the 5-bit gain input before it is latched to the parallel output pins.

B. Automatic Frame Synchronization

Our chip reads the biphasemark input stream looking for the start of a frame. Once a frame is detected our chip starts to decode the input stream. This means that our circuit can automatically synchronize with valid biphasemark data regardless of when the data stream starts, as long as the stream is valid. If the input is invalid then the output will be undetermined; once a valid frame is detected, correct decoding of the input stream will resume.

C. Validity Detection

Our chip reads the validity bit so that it does not send non-audio data to the output pins when the validity bit is set high.

D. Variable Sample Rates

Because our system clock is derived from the biphasemark input clock, our chip can be operated at different speeds. This means that our device can be used as a digital amplifier with a serial input and parallel output on any data stream that is encoded according to the IEC-986 format.

III. BASIC DESIGN AND COMPONENTS

Our chip design is relatively straightforward. Our chip first checks for frame preambles to properly align the data. It then decodes the biphasemark data and multiplies it to adjust the gain; it then uses a 6-bit counter to latch the correct audio data into one of two 21-bit registers, which contain the left and right output signals. Figure 5 shows a block diagram of our integrated circuit. The following sections provide details about the individual components.

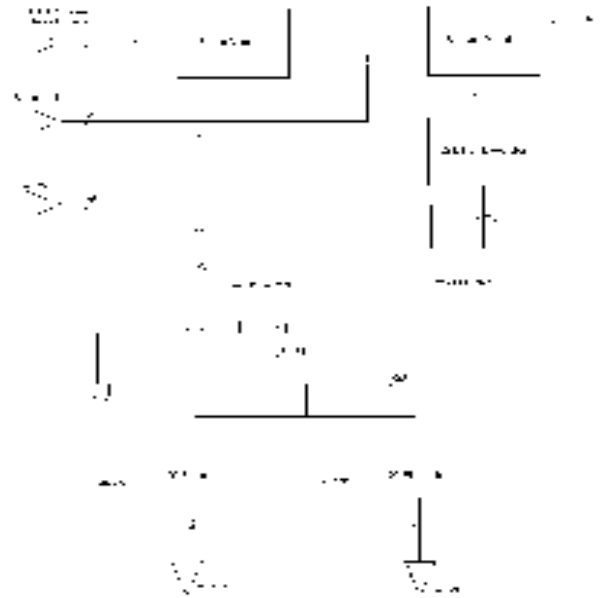


Fig. 5. This figure shows the basic block diagram of our integrated circuit.

A. Frame Detector

Our system uses a very simple method to decode the IEC-986 stream. When biphasemark data enters through the biphasemark input stream pin, it is shifted through an 8-bit shift register while searching for a specific pattern of bits that indicate one of two preambles that signify the start of a frame. The third type of preamble marks the start of a sub-frame so our circuit ignores these. Because only the preambles violate the biphasemark encoding, other data cannot be mistaken as a preamble. When the frame detector circuit finds one of the two preambles, our control circuitry is reset. The reason for this will be explained in a later section.

B. Biphasemark Decoder

After passing through the frame detector circuit, the biphasemark input stream goes into the biphasemark decoder circuit. This circuit shifts two values into a 2-bit shift register and latches the XOR of the two inputs into an output flip-flop at every alternate clock cycle; this result is the decoded value. The biphasemark decoder also produces an output signal that clocks the rest of the circuit. This signal is twice as long as the input clock signal so a simple one-bit counter is used to half the frequency of the input clock.

C. Shift Register & Array Multiplier

Of the 32 bits per sub-frame input by the shift register, only 16 bits contain the actual audio. These bits are input to the digital gain control unit, implemented with an array multiplier. The other operand to the multiplier is the 5-bit value from the gain input. The 16 by five bit array multiplier outputs a product of 21 bits. The multiplier is built using full adders and logic AND gates. This design implements a parallel multiplier based on the fact that partial products can be computed independently and simultaneously.

Array multipliers have a major disadvantage of taking up a lot of space, especially with large inputs. In order to save space, a more ideal approach might have been to use a Booth encoding algorithm. However the decoding and encoding overhead might have increased the delay of the operation, forcing us to decrease our clock speed. In the end we decided that with one 5-bit input, the array multiplier would not be too large to implement in our design.

D. Control Unit and Output Latches

Because a frame consists of 64 bits of data we decided to use a 6-bit counter as a control unit. When the frame detector finds the start of a frame, it resets this counter. This is to ensure that we latch the output from the multiplier into the correct 21-bit register at the correct time.

The control unit is therefore made up of a series of six full adders forming a 6-bit counter. On each falling edge of the clock, the 6-bit output of the counter is latched into six flip-flops. On the rising edge, the output of the counter is driven through the flip-flops.

The function of the control unit is to enable either the right or left channel to latch the 21-bit audio output from the multiplier. Because the input stream is fixed into 32-bit blocks, the output registers should only be enabled at the end of each block, at which time all of the data has been input. Thirty-two counts indicate the data is fed to the left channel, and 64 counts indicate the data goes to the right channel.

However, in addition to this counter signal, the output channels are enabled only if the validity bit is set low. This ensures that only valid data is latched and sent to the output. The output registers are

implemented using 21-bit flip-flops whose outputs are connected to output pins.

IV. INTERFACE

Our chip has a simple interface. Two pins are necessary to control the IEC-986 stream. The pin SIN takes the IEC-986 data signal, the pin CLK is used to define the biphase-mark data bit boundaries. The frequency of this clock signal depends on the sampling rate of the audio data. Because each frame contains a complete audio sample, and there are 64 bits in a frame, and the data is biphase-mark encoded, the clock frequency is equal to the sample rate times 128. For example, compact disk audio is sampled at 44.1 kHz, so the CLK signal should have a frequency of 5.64 MHz. The digital gain control bits are represented as a 5-bit unsigned integer using positive logic. Any change to these inputs will affect the most recently entered sub-frame.

The output of our chip is two 21-bit digital audio signals. These signals may be interfaced with additional digital logic or sent to a DAC to complete the audio conversion. We also route the biphase-mark-decoded signal to pin 73. A complete pin-out table is listed at the end of this paper for reference.

V. OPERATION

Our chip can be interfaced with many different audio streams yet some external circuitry is required to do so. Because our chip requires CMOS level input the IEC-986 voltage levels need to be converted to CMOS compatible levels. Because the differences between the IEC-986 format and the AES/EBU consumer format have to do with cabling and analog signal values, our chip can also be interfaced with AES/EBU consumer type input. As long as the digital input to our chip is parsed according to the IEC-986 format, our chip can be used. When interfacing our chip to an audio signal such as a CD player, the user must consider two things. First the user must ensure that the input clock signal is equal to the sampling rate times 128, and that the biphase-mark data matches the clock signal. One of the benefits of biphase-mark encoding is that because the polarity changes every even bit boundary, the clock signal can be

reconstructed from this signal. The user of our circuit needs to provide this clock signal. When designing a circuit to interface with our chip the designer should ensure that the input impedance of the circuit matches the impedance of the cable used to transmit the signal to the circuit. For example, if the user were interfacing our chip with the AES/EBU standard, the user should ensure that the input impedance is about 110 Ω (or a portion of the input energy will be reflected back to the source), and that +10 V high +3 V low maps to +5 V high 0 V low CMOS levels. Finally slew rate must be taken into consideration, as the signal frequency can be as high as 5.6 MHz. Interfacing our chip to a digital-to-analog converter is trivial as there are many inexpensive commercial digital-to-analog converters that accept CMOS level inputs on the market. Figure 6 shows how our device can be constructed to form a complete audio decoder.

TABLE 1.
COMMON INTERFACES

	AES/EBU	IEC-958
Cabling	110 Ω Shielded TP	75 Ω Coaxial or Fiber
Connector	3-pin XLR	RCA (or BNC)
Signal Level	+3 V Low +10V High	+0.5 V Low +1V High

VI. TESTING

The testing of our circuit began at the Standard Cell level. Each standard cell was tested using Verilog-XL at the transistor level. Following a successful simulation, the cell was created using Virtuoso in layout form. The layout was then verified by DRC and LVS. Once a successful LVS was run, an analog-extracted version of each cell was tested using the Analog Environment tool. These tests provided all the information used to create the “.lib” file used for our project.

After creating our simple standard cell library we

used these cells to create the major components of our design. We tested these components using Verilog-XL, DRC, and LVS. With more time, each cell would have been fully analog simulated.

Our chip passed initial Verilog-XL simulations with and without pads, we then wrote a program to generate a very large Verilog test program in Java, it did not work. As of now we are not sure if the error is in our generated test program, or our design, we suspect that the error is in our test script because our design passes a simpler handwritten Verilog test program. We also did a chip wide analog simulation for one complete audio frame and were disappointed to find that we did not get the expected output; we believe the error is an analog failure of the multiplier.

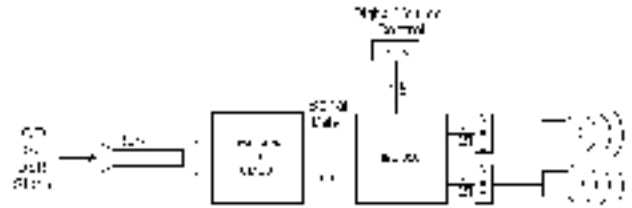


Fig. 6. Interfacing our device for audio conversion.

VII. CONCLUSION

We initially intended to create an on-chip digital-to-analog converter, we completed the schematic level design and the layout of the power transistors used in the amplifier circuit, but abandoned this goal for two reasons. One was that we ran out of time, and the other was that we would have had to put two of them on the chip (one for each channel), which would have taken up too much chip space. The work done on the digital-to-analog converter is listed in the technical report. We believe that even though our chip has not passed all simulations, our general design is good, and if given a little more time, could have produced a great chip.

