A Low Latency Multichannel Audio Interface for Low Power SIMD Digital Signal Processors

Lukas Gerlach, Guillermo Payá-Vayá, and Holger Blume
Cluster of Excellence Hearing4all, Institute of Microelectronic Systems
Leibniz Universität Hannover, Appelstr. 4, 30167 Hannover, Germany
Email: {gerlach, guipvava, blume}@ims.uni-hannover.de

Abstract—In this work, a first in, first out (FIFO) memory based architecture for a multichannel audio interface is presented. The architecture consists of an array of FIFOs, which is controlled to buffer and serialize multiple audio streams for interfacing other audio devices. This audio interface supports commonly used single instruction multiple data (SIMD) suited vector data formats. The load of SIMD processors for transferring audio data to and from the audio interface is therefore minimized. For this architecture, the processor load stays under 1% for transferring the audio data, whereas the load for related architectures rises up to 55% under certain conditions for the same workload. In order to decrease the power consumption, the FIFO addressing mechanism is used to set the processor to a low power state, if no audio data is present to be processed. Apart from the FIFO memory architecture, no further buffers, DMAs or interrupts are required. Compared to related architectures, the audio latency is significantly lower, since double buffering for DMA transfers is not required with the proposed architecture.

I. INTRODUCTION

Digital audio processing is one of the most demanding application for embedded systems like mobile phones, hearing aids and other multimedia devices. Due to the ongoing development of audio applications and the emergence of new application fields, there is the need for more processing performance, more audio channels, less latency, higher signal quality and longer battery life. Digital signal processors (DSPs) and application-specific instruction-set processors (ASIPs), which have been developed for these applications, are therefore highly optimized for performance and low power consumption. One approved and often used mechanism implemented in those processors is single instruction multiple data (SIMD), which enables efficient parallel processing of audio data. This mechanism increases the performance and efficiency in terms of energy.

For audio processing systems, an audio hardware interface is needed. This hardware is used to establish interfaces to analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) or is used as a serial communication interface. These communication interfaces transfer serialized audio streams to and from external devices, which can either be external audio codecs for the digital-to-analog and analog-to-digital conversion, other processors of the embedded system or wireless communication interfaces like Bluetooth devices. These external devices are not integrated on the same chip in many systems, since digital and analog circuits can not be implemented on the same die effectively [1].

In contrast to the optimized processors, the existing hardware interfaces for streaming audio data to and from external devices do not support the vector data types used for SIMD. Hence, audio data has to be permuted and aligned by these processors. As a result, the performance is decreased. Additional drawbacks in terms of performance and latency are caused by multiple buffers, direct memory access (DMA) units and interrupts.

This work presents an architecture for a multichannel audio interface, which is, like the processors, also optimized for performance, low power and low latency. Compared to related audio interfaces, the presented audio interface does support the SIMD suited vector data format for audio data. This implementation avoids an overhead of load for the SIMD processor caused by permutations of the audio data. The FIFO-based mechanism of this design does not need additional DMAs, buffers or interrupt controllers. The audio latency and the load for the processor can therefore be decreased, because less audio buffers are used and no interrupt routines have to be processed. The presented FIFO-based mechanism is also used to decrease the power consumption of the processor, if no samples are ready to be processed. Simulations and measurements are used to validate, evaluate and compare the proposed architecture to existing hardware interfaces.

This paper is organized as follows. In Section II, the related work is discussed. Based on the related work, the contribution of this work is described in Section III. The proposed multichannel audio interface architecture is depicted in Section IV. The results of the simulations and measurements are presented in form of a case study in Section V. A conclusion is given in Section VI.

II. RELATED WORK

In this section, the existing related audio processing systems are presented. These systems consists of a processing core and an audio interface, which is coupled to the processing core.

The TMS320C6747 [2] and the TMS320C6748 [3] are both fixed and floating-point digital signal processors, which are optimized for low-power applications. Both processors are equipped with audio hardware interface, which is called Multichannel Audio Serial Port (McASP) [4]. This interface consists of shift registers used for serialization (XRSR) of the in- and outgoing serialized audio data streams. If one sample is completely transferred, it is copied to or from an additional
register (XRBUF). This register can then either be accessed by the processor or by the DMA unit. Additional FIFO buffers can be activated when using the DMA to increase the tolerance of DMA latencies [2]. The synchronization is either done by interrupts or polling.

The Symphony DSP56300 family [5] are high performance multi-core digital signal processors. These processors are equipped with an audio interface called Enhanced Serial Audio Interface (ESAI) [6]. For serialization, this interface uses a transmit and a receive ESAI Shift Registers. If one slot is complete, the data of these shift registers is transferred to one of the ESAI Data Registers (RX0, RX1, RX2, RX3). The audio data stored in the data registers is written and read by the processor or DMA using different interrupts for different slots.

The ultra-low power CoolFlux DSP [7] provides shared FIFO memories, which are feed by DMA. The audio data can be transferred with the DMA to and from DAC, ADC and serial audio interfaces.

The Blackfin embedded processor family [8] is equipped with buffered serial ports (SPORT) and DMAs to transfer audio data. Two DMA channels are used for transmitting and receiving data and interrupts are generated every time a transfer is completed [9].

The Kinetis processor family [10]–[12] and the ADSP-21161 SIMD Sharc DSP [13] use standard DMAs and interrupts for transferring audio data to and from the audio interface and the local memory.

III. CONTRIBUTION OF THIS WORK

In this section the contribution of this work is introduced based on the architectures of the related audio interfaces. Some disadvantages of these architectures, which may in particular arise for audio processing, are discussed first.

Although most of the processors [2], [3], [7], [8], [10], [13], which are optimized for audio processing, support SIMD, none of the coupled audio interfaces support the SIMD suited vector data formats. As a result the processing performance of these systems is decreased, since the audio data has to be permuted by the processors every time that audio data has to be transferred to and from the audio hardware interface. This performance decrease is not negligible, since this task has to be performed repeatedly for every sample or audio frame. The frequency of this task depends on the sampling frequency.

Other performance degrading effects may be caused by interrupt based transfers. Interrupts occurring with the sampling frequency cause high interrupt rates. Even if the actual transfer is completely managed by the DMA itself, the processor has to process the interrupt service routine (ISR) on every interrupt, which generates execution load overhead for the processor.

Another disadvantage may be a high DMA activity of different DMA channels. DMA channels are typically serviced in a sequential order or may be serviced based on priority. High activity on other channels may therefore delay the transferring of audio samples and may therefore also delay the processing of these.

Latency may be added by using DMA transfers. Double buffering, also called ping-pong buffering, is used in the related work for DMA transfers in order to transfer and process audio data at the same time. In this case, the hardware related latency adds up to the software related latency. Ping-pong buffering doubles the latency proportional to the size of the buffer. Additional latency occurs where minimal size of transfers is limited or FIFO buffers are used additionally to DMAs to increase the tolerance for DMA latencies. These effects have been studied in [14]. The smallest feasible configuration of 16 audio samples for a DMA transfer restricts the latency in case of ping-pong buffering to 0.73 ms for a sampling frequency of 44.1 kHz. The additional FIFO buffers of 64 words increases the total latency by about 1.5 ms.

No related architectures provide low power mechanisms, which can be activated by the audio interface. The audio interface may not only trigger transfers or processing of audio samples but also trigger a low power state in the case when not enough samples are present to be transferred or processed.

To avoid the mentioned drawbacks, a new audio interface architecture is proposed in this paper. Compared to other existing architectures, this interface is based on a FIFO architecture. The FIFOs are implemented in hardware within the audio interface and the processor can access these directly using SIMD suited vector data formats. Additional DMAs or buffers are not needed. Instead of using interrupts for synchronization the processor is set to a special low-power mode, if it tries to read audio samples and no audio samples are there to be processed. This approach decreases the processor load, latency and power consumption compared to the related architectures.

IV. PROPOSED MULTICHANNEL AUDIO INTERFACE

The architecture of the proposed multichannel audio interface is presented in this section. This interface is connected to the internal processor bus and provides an serial inter-IC audio bus interface.

Since the audio interface is connected to either external audio devices or directly to digital-to-analog and analog-to-digital converters, the audio data has to be transmitted and received in a sequential order sample by sample for every audio channel. Therefore, audio interface has to serialize the data coming from the parallel processor bus and to parallelize the serialized audio data coming from the audio bus. The interface connections of such an audio interface on system level are shown in Figure 1.

![Fig. 1. Block diagram of a processor and the multichannel audio interface. An audio interface for on chip DACs or external devices can be provided. The standards for external inter-IC communication may be P^{S}, TDM or PDM.](image-url)
The serial audio bus width equals the resolution \( n \) of the DAC/ADC in case of a direct DAC/ADC connection and in case of an inter-IC audio serial bus the samples are transferred sequentially bit by bit. The standards [1] of this bus interfaces can be \( I^2S \) (inter-IC Sound), \( TDM \) (time-division multiplexing) or \( PDM \) (pulse-density modulation) for example.

In this case the parallel bus connection to the processor is 64-bit wide. Compared to other existing architectures [2], [3], [7], [8], [10], [13], this audio interface supports SIMD suited vector data formats. These formats are depicted in Figure 2.

In case of this 64-bit bus width, each word (64-bit) is either composed of 32-bit or 16-bit subwords. Every subword \( s \) represents one audio sample or time slot.

Samples stored within the subword of one word are of the same channel and sequentially ordered. Different audio channels can be read and written simultaneously by the processor within one clock cycle, which reduces the number of bus accesses needed to transfer the audio data. Since the audio data is already processed in a SIMD suited format by the processor, no additional permutations have to be performed.

FIFOs are used in this work to buffer the out- and ingoing audio samples, since the samples have to be transferred sequentially. The FIFO based memory access is preferred to a random memory access since a more efficient implementation and simpler accessibility can be realized [3], [7]. FIFOs, which are arranged and organized in an array as shown in Figure 3, can be used to support 64 bit read and writes as well as sequential accesses to single audio samples.

In the left part of Figure 3, a SIMD processor is shown accessing the multichannel audio interface. 64 bit of SIMD suited vector audio data can be written and read every cycle. Multiple audio channels can be implemented. The the least significant bits of the address bits are used to encode the audio channel. One row of the input or output FIFO array represents one audio channel. Depending on the address, all FIFOs of one of these rows are addressed. In case of a processor write access, 64 bits are written to four FIFOs of 16 bit width each. The width of 16 bit has been chosen for this implementation as the minimal supported width for a audio sample or time slot.

The output of the audio interface is a 1 bit data line for interfacing external devices using the \( I^2S \) or \( TDM \) standard. This implementation could also be used to interface an on-chip DAC/ADC by feeding samples from the FIFOs to the DAC/ADC. Depending on the bit clock line (BCLK) and the word clock line (LRCLK) of the \( I^2S \) standard, the data line is driven by the audio data stored in the FIFOs. This is done by the output control, which generates the FIFO address pointers and controls the multiplexer for the output of the FIFOs. Therefore, three multiplexer layers exist. The first multiplexer selects the channel in a time-division manner. For each channel the samples or time slots are sequentially select by the slot multiplexer from each FIFO in one row. The bit multiplexer serializes the sample and drives directly the output data line.

The same controlling unit is used for the serial input coming from external devices. A shift register is used to parallelize each sample or time slot. Depending on the channel and the time slot the audio data is then stored into one of the input FIFOs. The input FIFOs can then be read by the processor in the SIMD suited vector data format.

In order to implement a low power mechanism, the available FIFO address pointers of the FIFO-based audio interface architecture are used to check how many samples are available in the input FIFOs to be processed. If there are none and the processor is going to read audio samples from the audio interface, the processor is set to a low power mode to decrease power consumption. This mechanism is used for synchronization at the same time. The proposed approach was chosen instead of using interrupts, which is done in the existing implementations. The proposed approach therefore avoids high interrupt rates, which are caused by a continuous audio stream transfer, and hence also avoids a decrease in performance of the processor. Interrupts caused by other events are still possible depending on the underlying processor architecture.

In this work the KAVUAKA VLIW-SIMD processor presented in [15] is used to implement the proposed low power mechanism. The mechanism is shown in Figure 4. An idle flag is generated by the audio interface, which indicates that no audio samples are available in the input FIFOs. This flag is evaluated in the instruction fetch stage of the processor, if an \( IDLE \) instruction is used. This instruction is used before reading from the audio interface. If the idle flag is active and an \( IDLE \) operation is used, no operations (NOPs) are pushed into the pipeline of the processor. By this approach, interrupts caused by other events are still possible and the power consumption of the processor is reduced, till the switching activity within the processor is minimized.

A clock gating mechanism is an alternative mechanism to set the processor to a low power state using the idle flag. This option is not used, due to expected high silicon area requirement for routing the clock enable signal to every storage element of the processor. However, only a small amount of additional controlling logic is required with the proposed implementation of the low power mechanism for audio processing systems.
V. CASE STUDY: AUDIO STREAMING

In this section, the proposed multichannel audio interface is evaluated in a case study. The main goal of this case study is the comparison of different audio interface architectures in terms of latency and the performance impact on the processor. The latencies are determined by simulations and measurements using an audio analyzer. The measurement setup is shown in Figure 5.

The digital-to-digital latencies are determined by simulations and have been verified by measurements. The proposed multichannel audio interface is therefore coupled with a VLIW-SIMD processor presented in [15] as a case study. For the simulations, a hardware simulator is used. A model of a virtual audio codec is connected to the input and output of the audio interface. For the measurements the digital latency is determined by the audio analyzer, which is connected to the digital serial audio interface.

To determine and compare the latencies and performance impacts of different audio interface architectures a audio loop application is used for every test setup. This audio loop application transfers audio within the processor as fast as possible from the audio input of the audio interface to the audio output of the audio interface. No audio processing is done to avoid the impact of different processor performances. The CPU loads and latencies for transferring audio data are listed in Table I for different related audio interface architectures and the proposed FIFO based architecture.

The related architectures include either DMAs and FIFOs [2], [3], only DMAs [5], [7], [8], [10]–[12] or a direct processor bus interface to the registers of the audio interface, which
TABLE I
COMPARISON OF THE DIGITAL AUDIO LATENCIES AND CPU LOADS FOR PROCESSORS EQUIPPED WITH DIFFERENT AUDIO INTERFACE ARCHITECTURES.

<table>
<thead>
<tr>
<th>Audio Buffer Length</th>
<th>Processor</th>
<th>Audio Interface</th>
<th>DMA</th>
<th>FIFO</th>
<th>16 kHz</th>
<th>32 kHz</th>
<th>48 kHz</th>
<th>96 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>512</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>68.00 ms</td>
<td>34.00 ms</td>
<td>22.66 ms</td>
<td>11.33 ms</td>
</tr>
<tr>
<td></td>
<td>T1 TMS320C6747</td>
<td>yes</td>
<td>no</td>
<td>32.00 ms</td>
<td>12.07 %</td>
<td>16.00 ms</td>
<td>10.66 ms</td>
<td>5.33 ms</td>
</tr>
<tr>
<td></td>
<td>KAVUAKA</td>
<td>no</td>
<td>yes</td>
<td>32.12 ms</td>
<td>0.08 %</td>
<td>16.06 ms</td>
<td>10.70 ms</td>
<td>5.35 ms</td>
</tr>
<tr>
<td>256</td>
<td>T1 TMS320C6747</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>36.00 ms</td>
<td>18.00 ms</td>
<td>12.06 ms</td>
<td>6.00 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>yes</td>
<td>no</td>
<td>32.00 ms</td>
<td>12.07 %</td>
<td>16.00 ms</td>
<td>10.66 ms</td>
<td>5.33 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>no</td>
<td>yes</td>
<td>16.00 ms</td>
<td>0.08 %</td>
<td>8.06 ms</td>
<td>5.37 ms</td>
<td>2.68 ms</td>
</tr>
<tr>
<td></td>
<td>KAVUAKA</td>
<td>no</td>
<td>yes</td>
<td>16.12 ms</td>
<td>0.08 %</td>
<td>8.06 ms</td>
<td>5.37 ms</td>
<td>2.68 ms</td>
</tr>
<tr>
<td>128</td>
<td>T1 TMS320C6747</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>20.00 ms</td>
<td>10.00 ms</td>
<td>5.33 ms</td>
<td>2.66 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>yes</td>
<td>no</td>
<td>16.00 ms</td>
<td>12.07 %</td>
<td>8.00 ms</td>
<td>5.33 ms</td>
<td>2.66 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>no</td>
<td>yes</td>
<td>8.00 ms</td>
<td>12.07 %</td>
<td>4.00 ms</td>
<td>2.66 ms</td>
<td>1.33 ms</td>
</tr>
<tr>
<td></td>
<td>KAVUAKA</td>
<td>yes</td>
<td>yes</td>
<td>8.12 ms</td>
<td>0.08 %</td>
<td>4.06 ms</td>
<td>2.70 ms</td>
<td>1.35 ms</td>
</tr>
<tr>
<td>64</td>
<td>T1 TMS320C6747</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>12.00 ms</td>
<td>6.00 ms</td>
<td>4.00 ms</td>
<td>2.00 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>yes</td>
<td>no</td>
<td>8.00 ms</td>
<td>12.07 %</td>
<td>4.00 ms</td>
<td>2.66 ms</td>
<td>1.33 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>no</td>
<td>yes</td>
<td>4.00 ms</td>
<td>12.07 %</td>
<td>2.00 ms</td>
<td>1.33 ms</td>
<td>0.66 ms</td>
</tr>
<tr>
<td></td>
<td>KAVUAKA</td>
<td>no</td>
<td>yes</td>
<td>4.12 ms</td>
<td>0.08 %</td>
<td>2.06 ms</td>
<td>1.37 ms</td>
<td>0.68 ms</td>
</tr>
<tr>
<td>32</td>
<td>T1 TMS320C6747</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>8.10 ms</td>
<td>4.00 ms</td>
<td>2.66 ms</td>
<td>1.33 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>yes</td>
<td>no</td>
<td>4.00 ms</td>
<td>12.07 %</td>
<td>2.00 ms</td>
<td>1.33 ms</td>
<td>0.66 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>no</td>
<td>yes</td>
<td>2.00 ms</td>
<td>12.07 %</td>
<td>1.00 ms</td>
<td>0.66 ms</td>
<td>0.33 ms</td>
</tr>
<tr>
<td></td>
<td>KAVUAKA</td>
<td>no</td>
<td>yes</td>
<td>2.12 ms</td>
<td>0.08 %</td>
<td>1.06 ms</td>
<td>0.70 ms</td>
<td>0.35 ms</td>
</tr>
<tr>
<td>16</td>
<td>T1 TMS320C6747</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>6.00 ms</td>
<td>3.00 ms</td>
<td>2.00 ms</td>
<td>1.00 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>yes</td>
<td>no</td>
<td>2.00 ms</td>
<td>12.07 %</td>
<td>1.00 ms</td>
<td>0.66 ms</td>
<td>0.33 ms</td>
</tr>
<tr>
<td></td>
<td>TI TMS320C6747</td>
<td>no</td>
<td>yes</td>
<td>1.00 ms</td>
<td>12.07 %</td>
<td>0.50 ms</td>
<td>0.33 ms</td>
<td>0.16 ms</td>
</tr>
<tr>
<td></td>
<td>KAVUAKA</td>
<td>yes</td>
<td>yes</td>
<td>1.12 ms</td>
<td>0.08 %</td>
<td>0.56 ms</td>
<td>0.37 ms</td>
<td>0.18 ms</td>
</tr>
</tbody>
</table>

As shown in Table I, an audio interface equipped with a DMA and FIFO buffers results in the highest latencies for all audio buffer sizes and sampling frequencies. This is because the latencies caused by double buffering mechanism for DMA transfers adds up to the latency of the FIFO buffers, which store 64 audio samples each [14]. These additional FIFO buffers do not influence the processor load, but increase the tolerance of DMA latencies [2].

The processor load is defined here as the number of cycles needed for transferring the audio samples in one time interval divided by the total number of cycles available in this time interval. The clock speeds of the processors have been normalized for a fair comparison. If only DMAs are used for transferring audio samples the performance impact on the processor stays the same, but latencies are reduced by the latency of the FIFO buffers. The latencies can be drastically reduced by using only a single buffer for DMA transfers or by using no DMA. The direct processor accesses on every interrupt cause higher processor loads for bigger audio buffers but offer lower latencies compared to DMA transfers. In some cases, the processor load is smaller than using a DMA with double buffering, although the DMA interrupt rate is lower. The higher load is caused by the complexity of the DMA interrupt routine. In case of DMA transfers, semaphores have to be used to access the buffers while in case of the direct processor accesses this is not the case.

The smallest processor load is caused by the proposed FIFO based audio interface with SIMD vector audio data support. This is because no interrupts are used, the audio data does not have to be permuted and the processor can directly access

Fig. 5. Measurement setup for digital to digital and analog to analog latencies. The SIMD processor is connected by the audio interface to an external audio codec.

As is supported by all related architectures. All these interface approaches can be implemented within the TI TMS320C6747 processor [2], which is used in this case study as a reference architecture. The proposed audio interface with the FIFO based architecture is used with a SIMD processor presented in [15].

Different variations of parameters, which are relevant for audio processing systems, are investigated. Different sized buffers, which are needed for a frame based processing, are evaluated. The buffer sizes range from 16 to 512 audio samples. The size of 16 samples is in this case the minimal supported size for DMA transfers [14]. Different sampling frequencies are used to evaluate their performance impact, since higher sampling frequencies require higher data throughput. The evaluated sampling frequencies range from 16 to 96 kHz.
multiple samples. In this case, the latency is increased by two samples since two samples are stored within one 64 bit word, as is shown in Figure 2.

The analog-to-analog latencies are measured using the audio analyzer, which is in this case connected to the analog inputs and outputs of the used audio codec, as shown in Figure 5. A sine wave is generated at the input and the latency can be determined by monitoring the output of the audio codec. These waves are shown in Figure 6.

![Figure 6](image_url)

Table 6. Measurement of the analog-to-analog latencies for system verification.
The measurements are done using a audio analyzer connected to the analog in- and outputs of the external audio codec. The green (first) sine wave is generated by the audio analyzer and is used as an input for the audio codec. The yellow (second) sine wave is measured at the output of the audio codec. The latency is the time interval of these waves.

The results of the latencies are shown for two systems in Table II. The first system consists of the TMS320C6747 [3] processor and the TVL320AIC3106 audio codec. The audio samples are transferred using the DMAs without any FIFO buffering. The second system consists of the KAVUKA processor [15] and the ADAU1761 audio codec. The sampling frequency is set to the fixed value of 48 kHz for this measurement, since the latency of the audio codec varies with the change of the sampling frequencies. With increasing size of the audio buffer, the main part of the latency is caused by the digital audio interface. The system latency is comparatively smaller for the FIFO based architecture here, since no double buffering has to be used.

### Table II

<table>
<thead>
<tr>
<th>Buffer Length</th>
<th>TI TMS320C6747 + TVL320AIC3106</th>
<th>KAVUKA + ADAU1761</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>22.2 ms</td>
<td>11.68 ms</td>
</tr>
<tr>
<td>256</td>
<td>11.5 ms</td>
<td>6.35 ms</td>
</tr>
<tr>
<td>128</td>
<td>6.20 ms</td>
<td>3.68 ms</td>
</tr>
<tr>
<td>64</td>
<td>3.52 ms</td>
<td>2.35 ms</td>
</tr>
<tr>
<td>32</td>
<td>2.20 ms</td>
<td>1.68 ms</td>
</tr>
<tr>
<td>16</td>
<td>1.52 ms</td>
<td>1.35 ms</td>
</tr>
<tr>
<td>8</td>
<td>1.20 ms</td>
<td>1.18 ms</td>
</tr>
</tbody>
</table>

**VI. CONCLUSION**

In this paper, a low latency multichannel audio interface for low power SIMD digital signal processors is presented. The architecture is FIFO based and supports commonly used SIMD vector data formats. Compared to related architectures, which make use of DMAs, no double buffering and interrupts are used. Since no processing overhead is required to perform data permutations and no additional buffers are needed, the load of the coupled processor is reduced while the latencies are comparatively low. A low power mechanism based on the FIFO based architecture is presented in this work, which can set the processor to a low power mode, if no audio samples are ready to be processed.

**REFERENCES**


