Scalable microphone array system optimized for streaming inverse acoustics

J.A. Kauffman
University of Twente, The Netherlands, Email: j.a.kauffman@utwente.nl

Abstract

Sound source localization by using large microphone arrays commonly requires expensive hardware and an impractical large amount of cables and connectors. Our aim is to develop a compact and low-cost acquisition system that is expandable to hundreds of channels. Our design consists of a ring network of multiple sensor modules, each equipped with 16 input channels. One of the sensor modules interfaces with a computer through USB and functions as a master. Standard UTP cables are used to link the modules together for data transport and power supply. This principle reduces both weight and cost of cables. To increase streaming data rates, multiple ring networks can be connected to a computer to enable further expansion of the array. As the raw data stream is proportional to the number of elements in the array, data rates can exceed the maximum bandwidth of standard computer interfaces and storage devices. To overcome this problem, each module has a buffer for up to 45 seconds of raw data storage. Additionally, each module carries a digital signal processor (DSP) for optional signal processing. In the current configuration we preprocess each channel in order to reduce data rates and enable inverse acoustics as a streaming application.

Introduction

Microphone arrays find their use in many applications that require measurements of sound fields. An advantage of arrays over scanning-probe methods is that multiple locations within a sound field can be measured synchronously and within a short amount of time. These aspects are particularly advantageous for acoustical measurements involving non-stationary sources.

A disadvantage of large microphone arrays is that they are costly with respect to hardware and computational effort. The required data bandwidth and the amount of hardware increase linearly with the number of microphones used. For very large arrays of over hundreds of channels, the costs grow even more rapidly, as the high data rates require expensive high-throughput interfaces and storage devices.

We aim to develop a compact and affordable modular system that can be used to build large acoustical sensor arrays. Our focus is on reducing the amount of different hardware, cables and power-supplies, in order to keep the size, weight and costs of the system low. By incorporating preprocessing and buffering capabilities in the modular hardware, the system does not require an interface with a high-end computer.

This work is part of an ongoing research project in collaboration with the Section Structural Dynamics and Acoustics of the University of Twente and the Section Dynamics and Control from the Eindhoven University of Technology. The shared goal is to develop a high resolution 'acoustic camere' for noise localization. The first group works on methods for improved numerical modeling and validation in inverse acoustics [1]. The second group focusses on improving measurement techniques and signal processing [2]. Our task within this project is to develop a system for fast and user-friendly data acquisition and data reduction methods.

At this moment this project is still ongoing and the system has not yet been fully implemented. This paper describes our system design and the results of the preliminary experiments that we have performed so far.

System Design

To realize an expandable microphone array, we have designed a system consisting of multiple modules which can be connected in a ring network. Each module connects to 16 sensors (microphones or Microflowns[3]) and is configured through the network. One of the modules acts as a *master module*, and interfaces with a computer via USB. The others are *slave modules* and only interface with the ring network. Each module is able to capture raw data from its connected sensors. This data can be stored in on-board memory, transmitted over the system's ring network, or processed by an on-board DSP.

Modules

The modules are the basic building blocks of the multichannel system. Each module consists of two printed circuit boards; a main board and a sensor interface board. The sensor interface board is specific for the type of sensors used, for instance for balanced or unbalanced microphones, digital microphones or Microflown probes [3]. If necessary, it can be equipped with pre-amplifiers, filters and analog to digital converters (ADCs). The sensor interface board connects digitally with the main board. The design of the main board is the same for each module. Its main components are an FPGA (Altera Cyclone EP1C12Q240C6), a DSP (Texas Instruments TMS320-C6713), a USB microcontroller (Cypress EZ-USB FX2) and several memory ICs for data storage. All these are of-the-shelf components.

The main function of a module is to acquire the audio samples from the ADCs and to put these into packets for transmission. Packets can be transmitted to a computer via a USB connection, or passed to another module

through the system's ring network. There are three different types of packets: raw data packets, streaming packets and command packets. Raw data packets are 512 bytes long and contain 250 16-bit samples of a single channel. The remaining 12 bytes are used for header information, containing information about the originating board and channel number, the counter value of the first sample, and a checksum for error detection. Streaming packets are similar to raw data packets, but have a higher priority than raw data packets. These are used for time critical streaming data. Command packets contain parameters for device configuration and control.

Figure 1 shows a functional block diagram of a module. It is divided into three segments with individual clock domains: a receiver part, a transmitter part and a USB interface. The receiver clock is generated from the input signal transmitted by another connected module. The receiver input data is deserialized, decoded and filtered to ensure data integrity. Subsequently it is transferred to the (stable) transmitter clock domain via a synchronization buffer. The sample data from the modules own sensor interface is also received and packaged in this clock domain. If a module is configured for preprocessing tasks, the resulting data will be packed in a high priority streaming packet. Within the master module, all outgoing data streams are transferred to the computer via its USB interface. Furthermore, in this configuration incoming packages (with commands and settings) can be received from the USB interface. These packages are handled or transmitted to the other modules in the network. If a module is configured as slave, the data will be encoded, serialized and transmitted to the next module in the network.

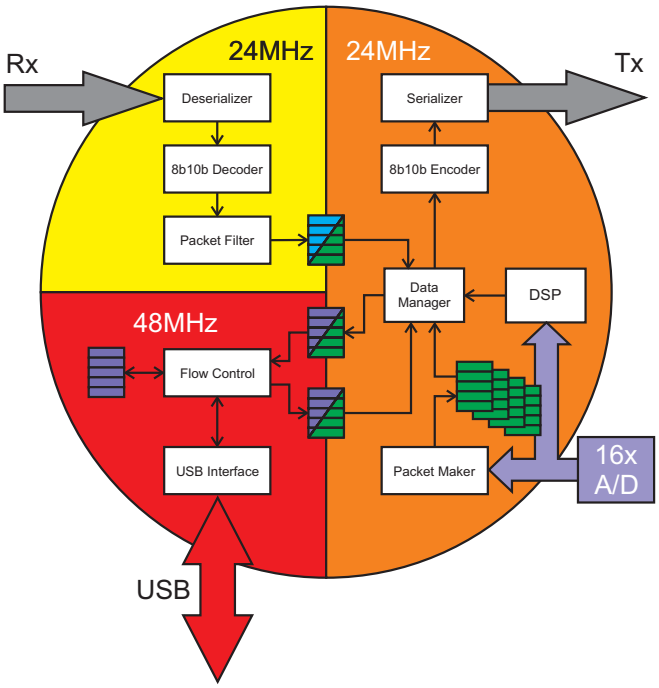

Figure 1: Data flow model in module. Three clock domains: receiver segment (Rx) locking on the transmitter clock of a sending slave node. Transmitter segment (Tx) with internal clock of 24MHz and a USB interface running at 48MHz.

If the data rate in the network exceeds the maximum data rate of the USB connection, the overflow of raw data packets are stored in on-board memory. The current design's memory buffer can store up to 45 seconds of raw data (16 channels, 16-bit samples at 48 kHz). The streaming and command packets have priority, and will therefore still be handled.

Topology

Multiple modules can be connected in a ring network using standard category 5 unshielded twisted pair (UTP) cables with 8P8C connectors (as commonly used for ethernet). Each module uses an independent clock for its transmitter, such that the number of concatenated modules is not limited by jitter build-up. The modules can be powered over the network by other modules, which reduces the required amount of power supplies and cabling. The required number of power supplies depends on the number of modules in the network and the power consumption of the connected sensors. Typically between 4 and 6 modules can share a single power supply.

By connecting one of the modules with the USB port of a computer, it automatically becomes the master module. All communications between modules are initiated and verified by this module. Data transmission takes place in one direction around the network; beginning and ending at the master module (Figure 2). A system initialization

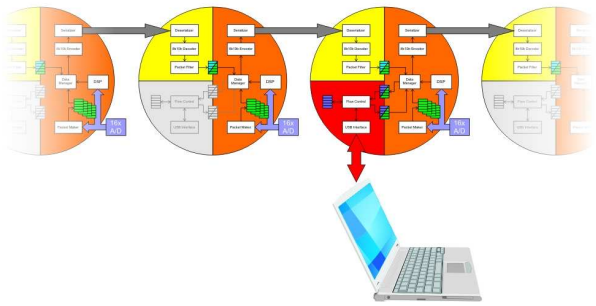

Figure 2: Ring-network of modules. The master module connects to a computer via a USB interface.

procedure is started by the computer software. First each connected module receives an identification number and reports back its number of available channels. Next, a synchronization cycle is started to synchronize the sample counters within each module. Subsequently, optional parameters can be set for preprocessing tasks. After initialization, the system waits for commands to begin the data acquisition.

The connections in the ring network consist of four separate transmission lines, each capable of 240 Mbit/s signal transfer. One line is used to transmit a word clock signal. The other three are available for data transmission. Since the communication lines are also used for (DC) power transfer, a DC balance must be achieved in the transmitted signals. We ensure this by applying an 8/10-bit encoding scheme [4], which adds two bits to every transmitted byte. Taking into account the overhead caused by the package headers and the encoding scheme, the maximum raw data rate

over the network's transmission lines is: $3 \times 240 \times (8/10) \times (500/512) = 562.5$ Mbit/s. This makes that the bottleneck is currently formed by the high speed USB connection, which is specified at 400 Mbit/s. However, in our own experiments we measured an average data rate of approximately 300 Mbit/s.

Using these figures, we estimate that a maximum of 400 channels (48 kHz, 16-bit) can be streamed to the computer. Beyond this number it is necessary to buffer the excess data in the modules, which limits the recording time to the amount of available memory.

Signal processing

An important feature of our design is that each module carries a DSP that can be programmed for preprocessing tasks. For instance, in NAH [5] the calculation of the acoustic hologram can take up considerable processing efforts. As this is a time-frequency transformation that can be performed independently for each channel, this task could be handled in the modules of the array. The general procedure would be to divide the time signals into segments of N samples and perform a fast Fourier transform (FFT) after applying a suitable window function. An overlap of 50% in the signal segements is often used to compensate for the information loss as a result of the windowing operation. Unfortunately these calculations lead to an increase of the amount of output data. This is not only due to the applied 50% overlap in the signal segments, but also due to the larger arithmic precision required for large FFTs [6]. Besides the increase of the data output, for streaming applications these calculations would require powerful processor hardware, which is not in line with our concept of an affordable and compact system. However, in many practical situations one may only be interested in just a single or a few frequency components. In such cases it is only necessary to acquire the data belonging to the desired frequency bin. Obviously this significantly reduces the data rate coming from the array. For such applications, it would not be necessary anymore to calculate the full FFT ($\propto N \log_2 N$ multiply-accumulate (MAC) operations), as it is more efficient to calculate the dot-product with the applicable discrete Fourier transform (DFT) kernel(s) ($\propto N$ MAC operations per frequency bin).

It can be calculated that for an array of 1024 channels sampling at 48kHz, with a single DFT kernel of 24000-points (1 Hz bins) and 50% overlapping segments, the streaming data rate is 256 kbps. This is a suitable data rate for streaming transfer to a computer where the final NAH calculations can be performed.

Software

When the array network is connected with a computer, a USB device driver needs to be installed first. The current driver is based on LibUSB, which is platform independent and open source software [7]. After installation of the driver, the device can be accessed by our development software.

Our development software is written in C# and makes use of multi-threading functions to enable semi-real time streaming applications. The software can access three different USB endpoints of the array device: (1) a control endpoint for sending commands and settings to the device, (2) a bulk data endpoint for reading raw samples from the memory on the modules, and (3) a streaming endpoint for reading time-critical data that has been preprocessed in the modules.

Data can be stored to memory or disk, or passed to a secondary thread or process for further processing. The current implementation also has an interface to Matlab (`http://www.mathworks.com`), which provides a practical platform for algorithm development and data visualization.

Preliminary experiments

To test our system, we constructed a four-by-four microphone array of 16 channels with a horizontal and vertical spacing of 15 cm. The used microphone probes are omni-directional, consisting of a small printed circuit board with an electret microphone (60 dB SNR, -42 dBV/Pa sensitivity) and a pre-amplifier with a line-level balanced output (calibrated at 1 Vpp for 94 dB SPL). This circuit is fitted in a brass tube, which is attached to an adjustable antenna shaped mount as depicted in Figure 3. Additionally, a small USB camera is attached in the center of the array for visualization purposes.

Figure 3: Four by four microphone test array with a camera in the center. The depicted microphone spacing is 15 cm.

The 16 microphones connect to the sensor interface board of a slave module. On this board, the analog signals are first low-pass filtered (20 kHz cut-off) to prevent aliasing effects during A/D conversion. The ADCs sample at 48 kHz and provide 16-bits for quantization. A second module connects to the slave module (by means of the ring network), and interfaces with a computer via USB.

To demonstrate our system, we implemented a streaming delay-and-sum beamforming [8] method in Matlab. Figure 4 displays a screen-shot of this demo. The

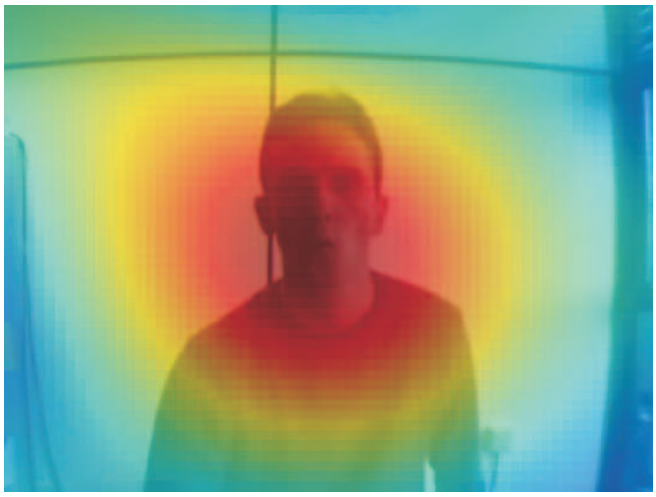

Figure 4: Beamforming demonstration: whistling at approximately one meter distance from the array. The red regions indicate large pressure signal amplitudes, the blue regions indicate small amplitudes.

calculations for this demo were performed on the raw data from the 16 input channels.

The described (streaming) preprocessing functionality was tested separately on a development evaluation kit of the selected DSP (TI TMS320-C6713). For 16 input channels, we found that it is possible to calculate the complex magnitude for up to 5 different frequency bins on a single DSP (running at 300 MHz).

Conclusions

Although our acoustical camera is still under development, we have been able to perform various tests to ensure the workings of the modular design. So far we have realized several identical reconfigurable modules that can be connected in a ring network. Also, we have been able to stream synchronous sample data from a 16 channel microphone array. As a proof of concept we demonstrated this in a beamforming application.

According to our experiments, the system's maximum (streaming) raw data rate is limited by its USB connection to approximately 400 channels. For larger configurations the excess data will be stored in on-board memory. This principle offers a practically endless expandability of the system. However, in such setups the recording time is limited by the memory capacity of the modules.

To enable semi-real time streaming applications for large configurations, we provide preprocessing functionality in each module. In case of NAH applications, where only a single or a few frequency bands have to be visualized, it is possible to reduce the data rate to a manageable level by calculating the acoustical holograms in the modules.

Acknowledgments

This work is supported by the Dutch technology foundation STW (Stichting Technische. Wetenschappen), project number TWO.6618 (inverse acoustics).

References

[1] J. Wind, M. Ellenbroek and A. de Boer, Moving sensors in inverse acoustics. In: 15th International Congress on Sound and Vibration, ICSV15, July 6-10 (2008)

[2] R. Scholte, Fourier based high-resolution near-field sound imaging, PhD. Thesis, Eindhoven University of Technology (2008)

[3] Reference to Microflown Technologies website. URL: `http://www.microflown.nl`

[4] A.X. Widmer and P. A. Franaszek, A DC-balanced, partitioned-block, 8B/10B transmission code, IBM Journal of Reasearch and Development, Vol. 27, No. 5, Page 440 (1983)

[5] Earl G. Williams, Fourier acoustics: Sound radiation and nearfield acoustical holography, Academic Press (1999)

[6] P.D. Welch, A fixed-point fast Fourier transform error analysis, IEEE Trans. Audio Electroacoustics 17: pp. 151–157 (1969)

[7] Reference to LibUSB website. URL: `http://libusb.wiki.sourceforge.net`

[8] M. Brandstein and D. Ward, Microphone arrays: signal processing techniques and applications, Springer (2001)