ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 4, April 2016

IMPLEMENTATION OF MULTIPLE AUDIO SWITCH USING DSP AND ALSA

Aman.J.Bhorwani[1]
[1]PG Scholar, Department of Electronics and Communication, R V College Of Engineering, Bangalore, India.

Chethana G.[2]
[2]Assistant Professor, Department of Electronics and Communication, R V College Of Engineering, Bangalore, India.

Bhargav Anur Krishnaprasad[3]
[3] Sr. Software Engineer Robert Bosch Engineering and Business Solutions Private Limited, Bangalore, India.

***Abstract*— The Multiple Audio Switch provides the capability for switching between three or more audio input terminals to participate in a multiple audio input conference. The Processor provides ability for the centralized processing of audio, video, and/or data streams. This processing can be mixing, filtering, switching, or other processing of media streams under the control of the multiple audio switch. The aim of this paper is to describe an audio processor based on Linux based sound driver software –ALSA and also using TI based DSP, for multiple audio switching. Method for implementing both the approaches are explained using pseudo code in this paper. Further as results and analysis, it gives a brief description about how proposed system is better using DSP rather than using ALSA.**

Keywords: Multimedia, DSP, Codec, ALSA, Multi-Channel Buffered Serial Port (McBSP), Application Programming Interface (APIs).

I. INTRODUCTION

A Multimedia System is characterized by the processing, storage, generation, manipulation and rendition of multimedia information. Recent development in the field of communication technology has increased the need for multimedia communication to large extent. Since telecommunication, video conference and many such advances in way of communication are pressing close to our lives, multimedia applications have also increased.

A Multimedia Application is an application which utilizes a gathering of numerous media sources e.g. content, representation, pictures, sound/sound, movement and/or video [1]. So here we propose one such application that switches between three such sound input terminals to output terminal. Here processor is an element in the framework which accommodates the incorporated preparation of sound, video or information stream in numerous sound exchanging. It gives blending, trade or other handling like separating under the control of various sound switch. Processor acquires sound, video or information stream from unified or blended different sound switch. Subsequent to preparing these media streams, it exchanges them to the assigned target terminal again under the control of the audio switch.

As the system takes real time audio input from the audio terminals, processing of the switched data by processor and again transferring it back to output terminal should be in real time. Dealing with real time data needs a processor fast enough to apply desired signal processing and write the processed data back to output terminal. Also the quality of the data should not be degraded, so appropriate processing is needed. Taking into consideration all such criterion the system resources used in the system plays an important role.

Usually for signal processing, DSP is considered to be appropriate due to its features which are very much suitable for signal processing applications. Choosing an appropriate DSP for our system is also an important matter of concern. Another way to implement our proposed concept can be through software framework and part of Linux kernel called Advanced Linux Sound Architecture; also known as ALSA. Thus this paper describes an audio processor based on Linux based sound card device driver –ALSA and also using TI based DSP, for a multiple audio switching. Further it also gives a brief description about how proposed system is better using DSP rather than using ALSA.

This paper is divided into different sections. Introduction section briefly gives overview of paper proposed. Second section explains the basic concept of multiple audio switch using block diagram. Following third and fourth sections gives overview of our concept implementation using ALSA and DSP respectively. Result section gives brief comparison

between both techniques used. Lastly the concluding section gives a better option among them.

II. SYSTEM OVERVIEW

Block diagram depicting basic concept of multiple audio switch is shown in Figure 1.

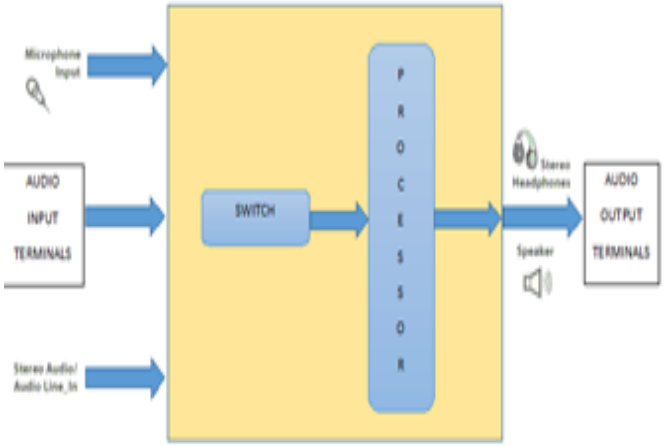

Figure 1: Block diagram of multiple audio switch

It can be seen from the block diagram that multiple audio input terminals can be interfaced to the system, using 3.5 mm audio jacks. This input audio can be routed as per needed with the help of toggle switch. This audio from selected audio source is passed on to processor block. The processor block performs operations like filtering of audio, amplification i.e. volume control, mixing of different audio and many such different sound effects. The processed audio from processor can be passed to audio output terminal.

Now the processor block shown in the block diagram, as mentioned in introduction section can be implemented using two approaches. One approach can be using Linux operating system based sound driver called Advanced Linux Sound Architecture (ALSA). ALSA can be used to mix, exchange, and filter and perform many such audio operations. All such operations in ALSA are done as different threads. Another approach is using an appropriate DSP that matches our processing requirement and perform all the processing operations using DSP. Both approach explained in following sections.

III. ALSA IMPLEMENTATION

ALSA stands for the Advanced Linux Sound Architecture. It consists of a set of kernel drivers, an application programming interface (API) library and utility programs for supporting sound under Linux. ALSA is a good choice if for performing low-level audio functions for maximum control and performance or want to make use of special features not supported by other sound APIs [5].

ALSA has a capability called plugins that allows extension to new devices, including virtual devices implemented entirely in software [4]. ALSA provides a number of command-line utilities, including a mixer, sound file player and tools for controlling special features of specific sound cards [8].

Below shown is the block diagram of ALSA architecture.

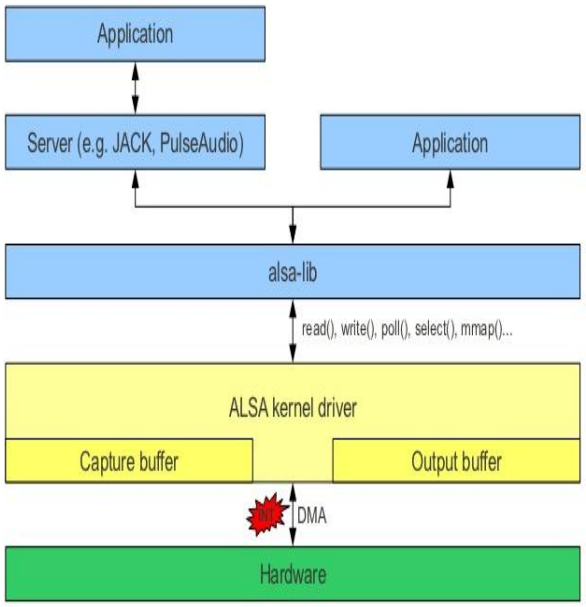

Figure 2: ALSA Architecture

This ALSA architecture is set in the system as main processing unit. Thus the ALSA application with the help of ALSA API's from device drivers, gets access to input as well as output hardware. It sets the necessary parameters of hardware to fetch data from input and output devices. Once hardware configuration is done, it performs different processing operations on data and writes it back to hardware.

There are two main modes namely, capture and playback in ALSA system [5]:

• Capture function is implemented by read data from the device (Mic-Reading)

• Playback function is implemented by writing data to such device (Speaker-Writing).

Pseudo code of ALSA in both the modes of operation can be given as

Main {

While (1) {

Getting access to soundcard device

Creating ALSA handle for access

Configuring different parameters

Reading/Writing to/from Device

Releasing the ALSA handle

Closing the access to soundcard device

}

}

It can be seen from the pseudo code that after including the soundcard header file, using the ALSA API's the device connected to system sound card is opened and configured. Next, parameters like number of channels, sample format, sampling rate, volume, etc. are set. Lastly according to the mode in use i.e. Capture/Playback mode, data bits are either read from connected device (capture mode) or written to device (playback mode) [6].

Below is the figure showing basic working of ALSA.

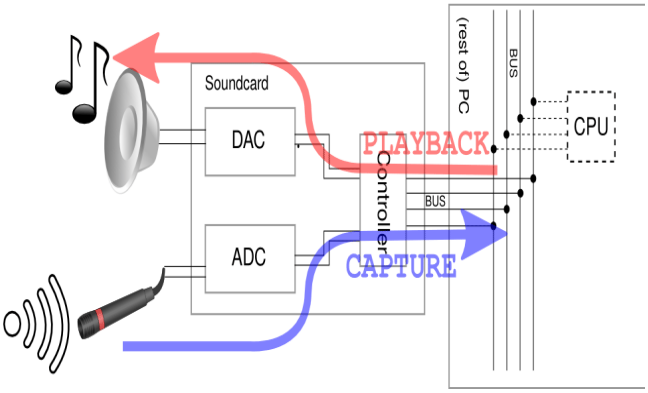

Figure 3: Modes of ALSA

Here we need a buffer to store the data bytes recorded from device by capture module through ADC of soundcard and pass it through processing module through CPU to perform desired operations. This processed data bytes are again stored in some buffer and passed it through playback module through DAC of soundcard to write it back to device. In order to perform all such operation in real time there needs to be some kind of synchronization between all three files to have access to the buffers. Thus to have synchronization between three functions, these are implemented as three different threads and semaphore concept is used to have synchronization between them.Hence ALSA is used as main processing unit in proposed multiple audio switch system.

IV. DSP IMPLEMENTATION

Digital Signal Processors (DSPs) are microprocessors with the following characteristics [2]:

a) Real-time digital signal processing capabilities. DSPs typically have to process data in real time, i.e., the correctness of the operation depends heavily on the time when the data processing is completed.

b) High throughput. DSPs can sustain processing of high-speed streaming data, such as audio and multimedia data processing.

c) Deterministic operation. The execution time of DSP programs can be foreseen accurately, thus guaranteeing a repeatable, desired performance.

d) Re-programmability by software. Different system behaviour might be obtained by re-coding the algorithm executed by the DSP instead of by hardware modifications.

Figure 4 shows a real-life DSP application, namely the use of a Texas Instruments (TI) DSP in a recorder–player.

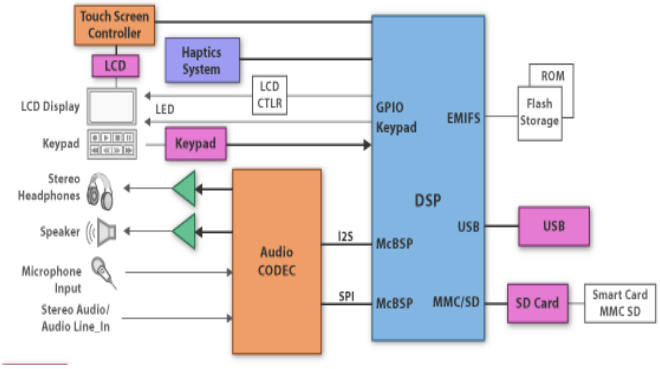

Figure 4**:** Use of Texas Instruments DSP in a player/recorder system [7].

Here since the input-output data is in analog form while the DSP processes in digital form, an audio codec is used to perform conversion form analog-to-digital and vice versa. The DSP implements the audio processing functions. Additional

tasks carried out are file management, user interface control, and post-processing algorithms such as equalization and bass management, filtering etc. Thus audio data from switch goes to codec and after getting converted to digital form undergoes required processing. From DSP it again passes to codec and after getting converted back to analog form is passed to output terminal. Also with the help of serial ports, GPIOS and memory interfaces external memory and other components like LCD, keyboard, etc. can be interfaced

Among varieties of DSP's and similar processors available in market form different sellers, Texas Instruments TMS320C5505 DSP is very much compatible with our system. Along with it, Texas Instruments AIC3204 codec is also used. The selection of Texas Instruments processor and codec is because of various factors including cost, system design, power, performance, support available, etc. but main reason is availability of Multi-Channel Buffered Serial Port (McBSP).The McBSP provides functions like Full-duplex communication , Double-buffered data registers, which allow a continuous data stream ,direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected analog-to-digital (A/D) and digital-to-analog (D/A) devices, external shift clock or an internal, programmable frequency shift clock for data transfer.

Using the Code Composer Studio IDE supported by TI DSP and the available support packages DSP and CODEC can be programmed for performing real time audio processing operations [3]. In this case, framework shown below can be used in our system.

Main {

While (1) {

Setting appropriate sampling rate

Setting ADC & DAC

Setting Line-In & Headphone-Out

Setting MIC & MIC-bias

Codec read

Performing processing operations

Codec write

}

}

Hence this way DSP is used along with a codec as a processing unit in our multiple audio switching system.

V. RESULTS AND ANALYSIS

Real time data can be recorded as well as played back in both the proposed techniques. All the required processes to be performed on the data can also be done by processor during the intermediate stage of capturing and playing back data. But both of the proposed techniques have their own pros and cons to be used for given multiple audio switching application. But considering given constraints of system application, cost, performance, resources and other similar parameters, a brief comparison is needed to come to a conclusion for choosing right processor for the system.

Considering the fact that ALSA is OS based sound device driver, it has limitations of dependence on operating system used. In case of any upgrades in the operating system in future, sound card driver also may get updated thus arises the need of regular upgradation of our application system. So is not the case with DSP. Once the appropriate code is burned to DSP, it may not need any future upgradations. Thus DSP is better in terms of maintenance than ALSA.

ALSA based application needs the system to be configured and necessary code implementing application also needs to be run on the system in case of change in system running OS, thus limiting application portability. While in the case of DSP implementation its hardware contains only DSP core and codec, it can be used anywhere regardless of change in system. Thus DSP is better in terms of system portability than ALSA.

In terms of flexibility of system, ALSA is better than DSP because since its OS based application necessary changes can be made in implementation at any time while for DSP once the code is flashed to DSP and hardware is configured it's difficult to make changes in future.

Considering comparison in terms of system resources, it is mainly dependent on the number of data channels to be used. Figure 6 has shown the occupied resources in ALSA implementation go up with the increment of channel numbers. It can process the exchange of channels of audio data stream depending on the number of sound cards interfaced to the system. There is limitation to this number of interfacing depending on system. Also ALSA implementation uses multithreads and has the feature of owning its independent global variable, occupies large system resource. With DSP implementation, the resource of system increases slowly because DSP uses multichannel buffered serial port (McBSP) which can handle 4 codec interfacing, each again able to process 4 channels at a time. When processing 12 channels of

audio data stream exchange, it is still less than 20% in comparison to ALSA.

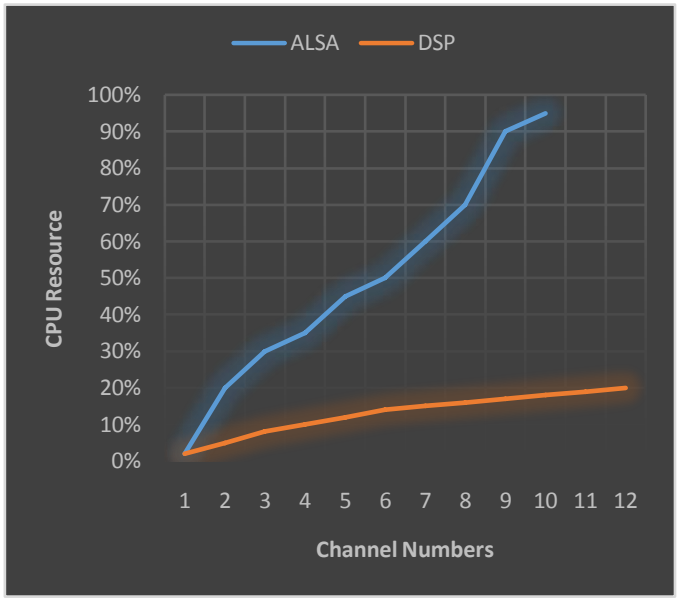

Figure 5: Comparison of system resource

Thus above graph gives a rough estimation of system resource needed with respect to number of channels for both ALSA and DSP implementation.

Another aspect of performance is better in DSP as it's meant for only specific data processing operations while ALSA being operating system based device driver is comparatively less in terms of performance. Also in terms of power, running DSP and codec requires less power than running whole operating system based machine. All the proposed experiments were done using TI TMS320C55X DSP, TI AIC3204 codec and Ubuntu 12.0.4 based ALSA 2.6 software framework.

VI. CONCLUSION

This paper introduced two techniques of multiple audio switching and processing application using ALSA and DSP. After a brief comparison between both of them in general terms of flexibility, portability, maintenance, system resource, power and performance, it can be concluded be that for given specific concept of multiple audio switching and processing, DSP is better than using Linux kernel based ALSA.

REFERENCES

[1] B. Zhang, "Design and implementation of a scalable system architecture for embedded multimedia terminal", Guilin, China, pg. 618 – 621, 2011

[2] R. G. M. Hilkens,"An educational DSP platform based on a TSM320C5505 EZDSP ," Eindhoven University of Technology Den Dolech 2, 5612 AZ, Eindhoven, The Netherlands, pg. 159 - 163, 2012.

[3] Song Wu,"TI DSP implementation of a medium speed DSL (MDSL) for multimedia applications", DSPS R&D Center, Texas Instruments. Inc., TX, USA, pg. 3149 - 3152 vol.5, 2008

[4] J. M. Lin, "Software Integration for Applications with Audio Stream," Feng Chia Univ., Taichung, pg. 1126 – 1129, 2008

[5] Introduction to Sound Programming with ALSA. [Online]. Available: http://www.linuxjournal.com/article/6735?page=0,0# [Accessed: 10-Dec-2015]

[6] Asynchronous playback, [Online].Available: http://alsa.opensrc.org/Asynchronous_Playback_%28Howto%29#Mandatory_Parameters. [Accessed: 10-Dec-2015]

[7] Picture courtesy of Texas Instruments from www.ti.com

[8] Advanced Linux Sound Architecture.[online].Available: https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture