Next Generation Audio System for Chrome OS using Sound Open Firmware

Submitted By

Smit Jayeshkumar Shah

17MCEN12

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Next Generation Audio System for Chrome OS using Sound Open Firmware

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering (Networking Technologies)

Submitted By

Smit Jayeshkumar Shah

(17MCEN12)

Guided By

Dr. Ankit Thakkar

Company Guide

Samaga Prasanna Krishna

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Certificate

This is to certify that the major project entitled **" Next Generation Audio System for Chrome OS using Sound Open Firmware "** submitted by **Smit Jayeshkumar Shah (Roll No: 17MCEN12)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Netwoking Technologies) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Ankit Thakkar
Guide & Associate Professor,
CSE Department,

Institute of Technology,
Nirma University, Ahmedabad.

Dr. Gaurang Raval
Associate Professor,
Coordinator M.Tech - CSE (Networking Technologies)
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Madhuri Bhavsar
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad.

Certificate

This is to certify that the major project entitled **" Next Generation Audio System for Chrome OS using Sound Open Firmware "** submitted by **Smit Jayeshkumar Shah (Roll No: 17MCEN12)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Netwoking Technologies) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Samaga Krishna Prasanna
Company Guide,
Software Engineering Manager,
Intel Corporation, Bangalore.

Statement of Originality

I, **Smit Jayeshkumar Shah**, **17MCEN12**, give undertaking that the Major Project entitled "**Next Generation Audio System for ChromeOS using Sound Open Firmware**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (Networking Technologies)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Dr. Ankit Thakkar

(Signature of Guide)

Statement of Originality

I, **Smit Jayeshkumar Shah**, **17MCEN12**, give undertaking that the Major Project entitled "**Next Generation Audio System for ChromeOS using Sound Open Firmware**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (Networking Technologies)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Samaga Prasanna Krishna

(Signature of Company Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Ankit Thakkar**, Associate Professor, Information and Technology Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Madhuri Bhavsar**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Science and Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

I am much obliged to **Samaga Krishna Prasanna**, Software Engineering Manager, Intel Corporation, Bangalore for his special attention, guidance throughout my Internship tenure. I would also like to thank my all team members who supported me and helped me in learning. A big thank you to Intel Corporation for giving me an immense opportunity to learn and grow in industry.

- Smit Jayeshkumar Shah

17MCEN12

Abstract

Chrome OS is an open-source project that aims to build an operating system that provides a fast, simple and more secure computing experience for people who spend most of their time on the web. Chrome OS uses the Linux kernel, it has a large pool of contributors. Audio drivers shipped in an open-source manner, proprietary DSP firmware will remain closed and shipped in binary. As a result, firmware issues become very difficult to address. Intel is partnering up with Linux Foundation and Google to come up with Sound Open Firmware project. By using SOF, developers may be able to debug the firmware issue more quickly. SOF will allow the developers to optimize footprint and performance by adding only functionality part in their DSP. SOF SDK has total five component, the firmware source code, firmware tools to convert firmware into appropriate formats and to debug, a toolchain for firmware image creation, an emulator to trace and debug drivers & firmware, and ASoC Linux kernel drivers. The SOF project provides capabilities for developers to innovate and enhance DSP functionality. As of now, SOF supports the Cadence Tensilica Xtensa instruction set architecture of DSP hardware and software. SOF project is licensed under BSD. This project contains the work carried out by me while integrating SOF driver to Intel platform based Chromebook.

Abbreviations

SOF	Sound Open Firmware.
ALSA	Advanced Linux Sound Architecture.
CRAS	Chrome Audio Server.
DSP	Digital Signal Processing.
SDK	Software Development Kit.
ARC	Application Runtime Container.
OS	Operating system.
DAI	Digital Audio Interface.
DMA	Direct Memory Access.
PCM	Pulse Code Modulation.
SoC	System on Chip.
PD	Platform Driver.
MD	Machine Driver.
IRQ	Interrupt.
MMU	Memory Management Unit.
Mux	Multiplexer.
PD	Platform Driver.
MD	Machine Driver.
IO	Input and Output.
DAPM	Dynamic Audio Power Management.
IPC	Inter Process Communication.
DAC	Digital to Analog Converter.
ADC	Analog to Digital Converter.
CPFE	Chrome Partner Front End.

Contents

Certificate iii

Certificate iv

Statement of Originality v

Statement of Originality vi

Acknowledgements vii

Abstract viii

Abbreviations ix

List of Figures xi

1 Introduction 1
- 1.1 What is Chrome OS ? . . . 1
- 1.2 Architecture of the Chrome OS . . . 1
- 1.3 Chrome OS Audio stack . . . 2
  - 1.3.1 CRAS - Chrome Audio Server . . . 2
  - 1.3.2 ALSA - Advanced Linux Sound Architecture . . . 3
  - 1.3.3 Other Main Terms . . . 4

2 Sound Open Firmware 6
- 2.1 Introduction to SOF . . . 6
  - 2.1.1 SOF SDK Component . . . 6
  - 2.1.2 SOF Architecture . . . 7
  - 2.1.3 SOF Driver Architecture . . . 8

3 Getting Started on Work 10
  - 3.0.1 Prerequisites . . . 10
  - 3.0.2 My Contribution and Work . . . 11

4 Conclusion 28
  - 4.0.1 Conclusion . . . 28

Bibliography 29

List of Figures

1.1 Chrome OS Architecture[1] 2
1.2 Chrome OS Audio Stack[2] 3
1.3 Chrome OS Audio Stack Internal 4

2.1 SOF Architecture[3] 7
2.2 SOF Driver Architecture[3] 9

3.1 Script to Automate the Porting 12
3.2 Ported Patch List 13
3.3 DMIC pop Noise Issue 15
3.4 Patch from audio maintainer's tree 23
3.5 Backport patch format 24

Chapter 1

Introduction

1.1 What is Chrome OS ?

Chrome OS is a browser-based operating system where it runs on Linux kernel. Chrome OS is introduced by Google in the year 2009. Google is optimizing Chrome OS by adding more features & capabilities. It has millions of user across the world. Its unique hardware and software security mechanisms make it one of the most secure operating system. It does not need a compute-intensive machine. It is mostly focused on the education system & consumer market but soon into the enterprise market.

Chrome OS is built using Chromium OS, Chromium OS is open source and available to all the developers and contributors, The difference between Chromium and Chrome OS, some of the key parts of the Chrome OS is not yet open source, Google add some of their secret ingredients to make it more optimized and secure. Chromium OS can be build using its source code, we can use the shell and the root access. Chrome OS is available pre-built and already installed in the Chromebook.

Chrome OS has capabilities to run android & Linux based applications and tools.

1.2 Architecture of the Chrome OS

Figure 1.1 is the architecture stack of the Chrome OS. It has a coreboot. Coreboot is a special lightweight firmware, designed to do a minimum task like loading the OS. It runs on the modified Linux kernel, configured for the specific platform. It uses different libraries like graphics, 3D, audio etc. Chrome browser does most of the work. At the

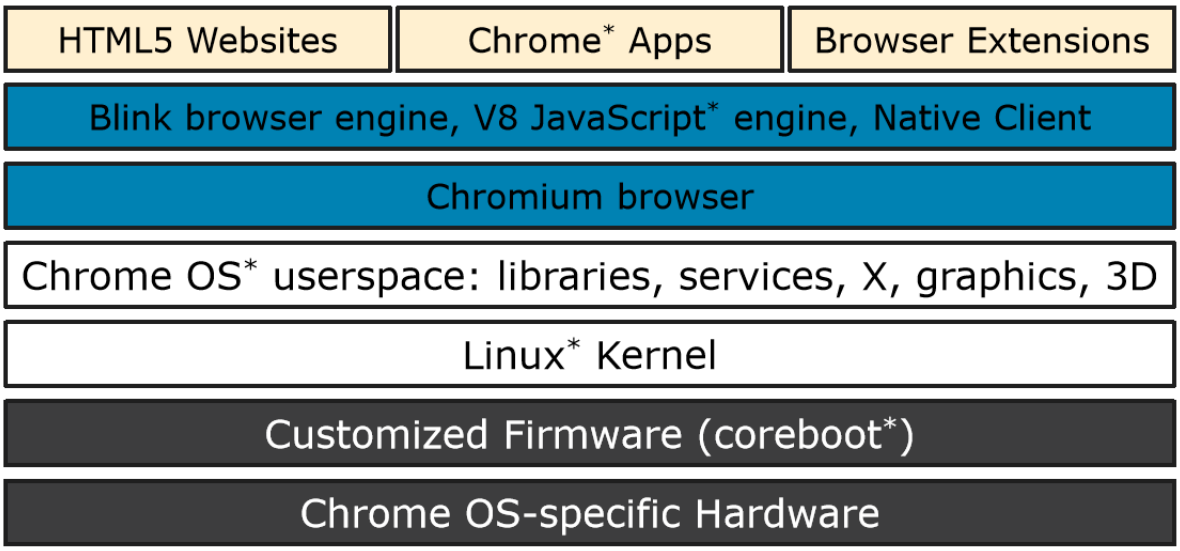

Figure 1.1: Chrome OS Architecture[1]

user end, chrome apps, websites, browser extension etc.

Chrome OS is one of the most secure operating system. It is a browser-based OS. All the domains and web apps are isolated from each other, Internet will make it more secure. The file system of the OS is designed in such a way that user's data will be encrypted on one partition while all root files are being stored on another partition. If it gets the OS update, user's data will not be impacted. It has two root partitions. In an update, only root partitions B will be updated, root partitions A will not be altered. If fails while update, a user can reboot the device so OS can load from root partition A. If no issue after update, we can flip the switch to root partition B. Apart from verifying the Google's signature on an image, It also keeps the kernel code secure. Upon every boot, OS will keep maintaining the hash code of each kernel code blocks, If any alteration is done in OS kernel code then the hash function will also get change. OS will get the kernel code tampering, a device will reboot, it can eliminate the malicious code. This feature calls the verified coreboot.

1.3 Chrome OS Audio stack

Figure 1.2 & 1.3 is Chrome OS audio stack. CRAS & ALSA are two main components.

1.3.1 CRAS - Chrome Audio Server

CRAS allows audio to route dynamically to the newly attached devices. If audio is routing to the device speaker, if hot-plug the headset then CRAS will route the audio to

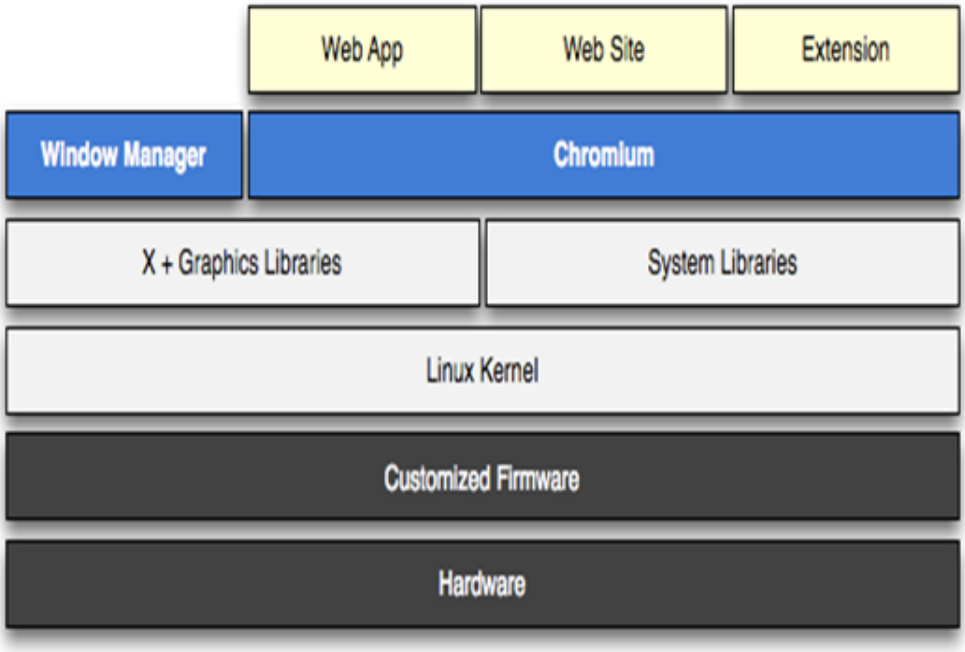

Figure 1.2: Chrome OS Audio Stack[2]
Source :- www.Chromium.org

the new device. If two device hot-plugged at the same time, routing will be decided by the priorities of device. It uses very low CPU usage and the latency of the 20ms. Audio data will be exchanged through the shared memory.

The browser handles decoding and CRAS handles the mixing. If in two different tabs, we are playing audio then CRAS will mix audio. CRAS will discover new devices. If a new device hot-plugged, by using interrupt it will adjust the routing. The volume control will be handled by the CRAS.

1.3.2 ALSA - Advanced Linux Sound Architecture

ALSA is the standard user interfaces for the user space application or the middleware. It supports all type of Audio interfaces from consumer sound card to a professional multichannel audio interface. It has a modularized sound driver which can be modified at the runtime. It contains the Kernel, mixer, sound file players, API libraries and plugins. It has the userspace library (ALSA-lib) to simplify the application programming and provide a higher level of functionality.

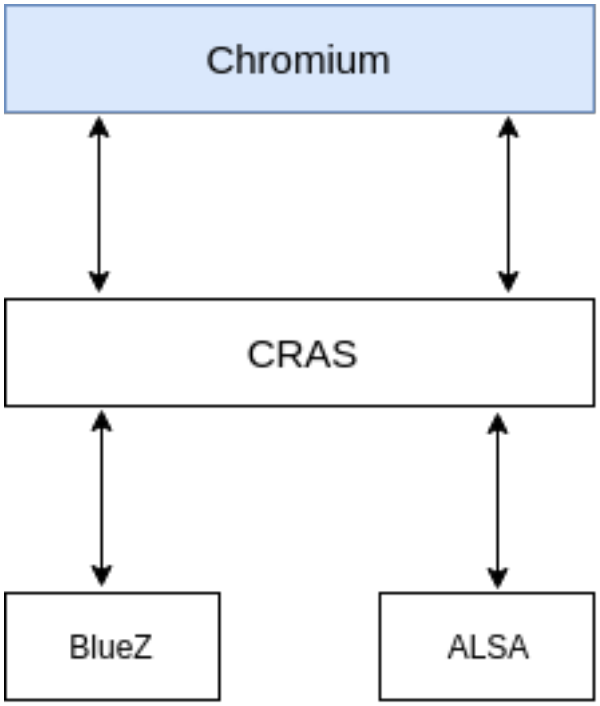

Figure 1.3: Chrome OS Audio Stack Internal

1.3.3 Other Main Terms

DAPM - Dynamic Audio Power Management

DAPM is one of a feature which helps CRAS to use the low CPU usage, It dynamically manage the power within an audio subsystem, developer doesn't need to recompile or change the code for userspace applications, DAPM manages power switching decisions based upon any audio functionality like capture or playback and audio mixer settings within the device.

Platform Driver

Platform driver describes the DAI and it supports. It configures the DMA for the data transfer. Platform driver configures the PCM port of the SoC. It ensures codec and SoC are configured correctly and matching then audio functionality can work properly.

Machine Driver

Machine driver register the sound card, binds platform driver and codec driver together. It handles the sound card events such as push button event. It needs the board specific driver and needs a different driver for each board configuration. Machine Driver can contain the codec and platform-specific code.

Codec Driver

It is a generic driver that maps the codec topology. It provides the interface for enabling the DAI in the codec side. Each element in the codec is a Widget. Widgets are interconnected using the map. All configurable parameters should be the user configurable. It is board independent and can reuse across a platform.

Chapter 2

Sound Open Firmware

2.1 Introduction to SOF

Audio driver shipped in an open-source manner, proprietary DSP firmware will remain closed and shipped in binary. As a result, firmware issues become very difficult to address. Intel is partnering up with Linux Foundation and Google to come up with Sound Open Firmware project. SOF will allow the developers to optimize footprint and performance by adding only functionality part in their DSP. SOF SDK has total five component, the firmware source code, firmware tools to convert firmware into appropriate formats and to debug, a toolchain for firmware image creation, an emulator to trace and debug drivers & firmware, and ASoC Linux kernel drivers. The SOF project provides capabilities for developers to innovate and enhance DSP functionality. As of now, SOF supports the Cadence Tensilica Xtensa instruction set architecture of DSP hardware and software. SOF project is licensed under BSD.

2.1.1 SOF SDK Component

Five main component of SOF SDK.

- SOF Source Code[3].
- SOF Tools[3].
- ASoC Linux Kernel Drivers[3].
- Crosstool-NG Tool-chain[3].
- Qemu DSP and Host Emulator[3].

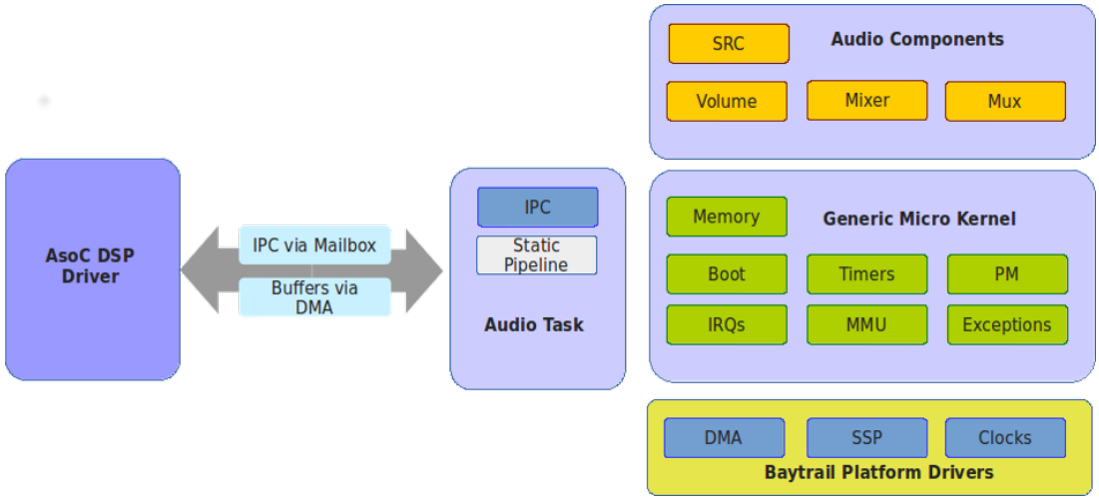

Figure 2.1: SOF Architecture[3]

1) SOF source code is well written in the C language with some architecture specific assembler. It does not link to any external dependencies.

2) SOF tools are required to convert the firmware from the ELF format to one, which understands by the kernel drivers and tools. It used to assist with debugging running firmware image.

3) ASoC Linux kernel driver is required to register the DSP firmware and to expose the PCM, KControls etc.

4) Cross tool-NG used to build a GNU cross toolchain like gcc, gdb, binutils etc, which used to build the firmware binaries. There are no restrictions to use this tool and compiler, most preferred tool among developer.

5) Qemu is used to provide a functional emulator to simultaneously trace and debug driver and DSP firmware code.

2.1.2 SOF Architecture

Figure 2.1 is the SOF Architecture where it has total three main component

Audio Component

In an audio component, there is a source, volume, Mixer and Mux. It is used to form an audio processing pipeline from the host DMA buffer to the DSP DAI. It has source and sink buffers to transform or route the audio data as a part of processing.

Generic Micro kernel

Generic microkernel manages and abstracts the DSP hardware for the system. It extracts all the information about the DSP and other hardware for the system. It exports C API for memory allocation, scheduling work, event notification, and power management. It has many small components which kernel handles like a boot, memory, IRQs, timers, MMU, exceptions etc.

Platform Drivers

Platform driver is used to control any external IP to the DSP IP. Platform driver includes DMA engines and DAI controllers. It is used by the audio components and pipelines to send/receive data to/from the host and external codecs. All process related to DMA, SSP, Clocks will be handled by the platform driver.

Audio Task

It handles the audio pipeline at run-time. It also handles the transportation of data from source to sink within the pipeline. In SOF pipelines are dynamic.

2.1.3 SOF Driver Architecture

Figure 2.2 is the SOF driver architecture. It has four main component.

- ASoC Machine driver[3].
- Generic PCM Driver[3].
- Generic IPC Driver[3].
- DSP Platform Driver[3].

ASoC Machine Driver

In the ASoC machine driver, It handles the codec integration, board integration and hardware configuration with the platform driver and system. All process related to codec & board integration will be handled the Machine driver. It binds the platform and the Codecs drivers.

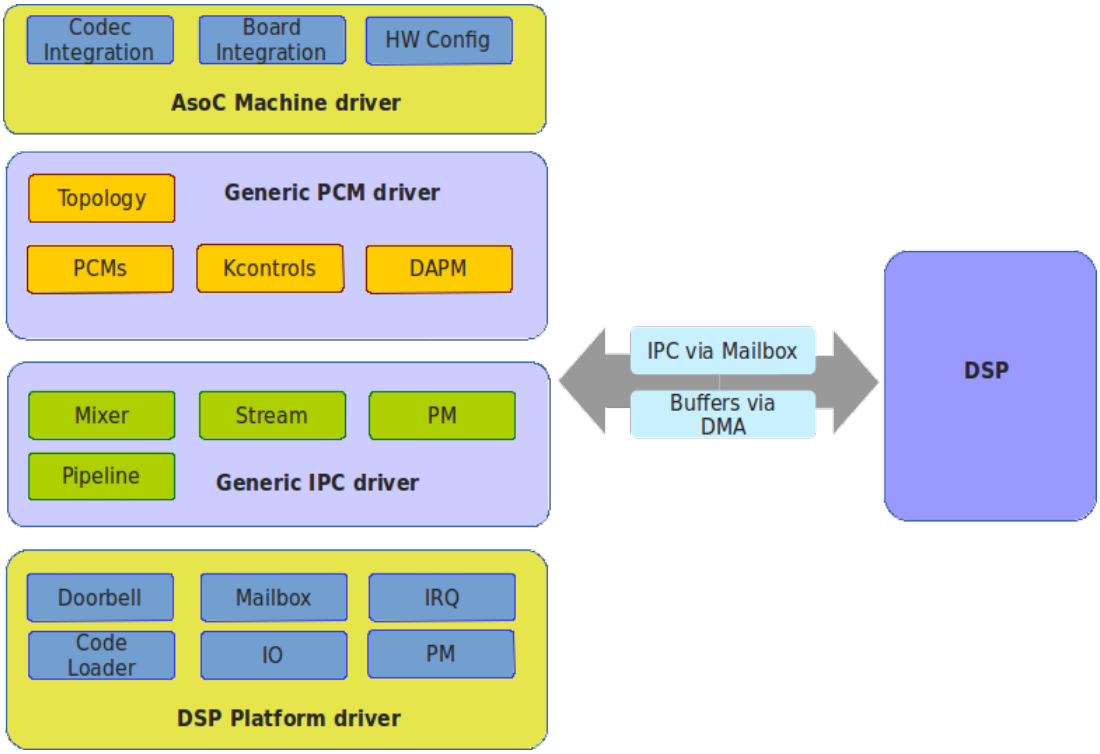

Figure 2.2: SOF Driver Architecture[3]

Generic PCM Driver

It has the topology which is architecture specific, configured board specific. It handles the channel selection for different playback. DAPM is a part of PCM driver which works in the power management within the Audio subsystem.

Generic IPC Driver

Mixer, stream and pipeline will be handled by the IPC Driver. SOF uses dynamic pipeline which will allow OS to effectively use the pipelines.

DSP Platform Driver

Platform driver handles hardware and software IRQ. Hot-plug event and input & output will be handled by the DSP platform driver. It also handles the input and output.

Chapter 3

Getting Started on Work

3.0.1 Prerequisites

I have joined Intel as a graduate technical intern in Chrome OS enabling team. The team is responsible for the integration & validation work on Intel platform based chromebook. Internal trainings had been conducted to give more detail about OS architecture, kernel development, audio, mandatory tools etc. Below is the detailed list of training.

- Setting up the Linux machine. High configuration system is required for Chrome OS development.
- Download and install required packages for development.
- Configuring the packages for development.
- Team uses different tools like git, Gerrit, JIRA. Git is a version control system. Gerrit is team collaboration tool and code review platform. JIRA is project & bug management tool.
- Configuring, building & flashing kernel.
- Building and installing Chromium OS.
- Updating coreboot with and without servo board.
- Pushing patches to different branches.
- Generating SOF firmware & topology file

3.0.2 My Contribution and Work

After different trainings, Initially I had been assigned to the lower priority issue so I can understand OS architecture and kernel flow in detail. In the subsection, detailed work has been explained which I had done during the internship.

(1) tty-S1 Failed to request DMA

Time Spent :- 1 Week

Issue Detail :- This issue is reported on JIRA. It was easy to reproduce, first I had to check on given CPFE image version. CPFE image is securely stored by the Google, only Google employee and their Chrome OS partners have an access to it. The procedure had been given on a JIRA to reproduce the issue. I had to reboot the Chromebook for multiple time without any peripheral device being connected to system. Developer mode in Chromebook has a feature to check the kernel logs and root files. After enabling the developer mode and checking kernel logs, it had this message tty-S1 Failed to request DMA.

Root Cause :- For root cause of issue, I had to check the serial driver code because tty-S1 is a serial port. I understand the code and find out the function with this message. This code block will get executed as a part of bootup initialization. This error message had no impact over the functionality of OS. The best practice is to keep the kernel message clean. I had enabled a few more parameters and put printk statements to understand the execution flow of code. After building and flashing the kernel on a system. It is just a warning. As a part of the bootup initialization process, the serial port will try to get initialized. Failing on 1st time it will print this warning message, not an error message. After retrying for another time it will get initialized.

Solution :- I had discussed this issue with the team and we had concluded it that, since it doesn't have any functionality impact in OS, We can close this issue as the messages in dmesg is only warnings.

(2) Classification and Submission of SOF & Legacy Audio patch

Time Spent :- 10 Days

Summary :- After SOF driver integration starts, Initially there was only one git master branch which has all the SOF driver and legacy audio patches, there were around 300

```
echo " Applying first set of patches from 1198727 to 1198728 "
i=27
while [ $i != 29 ]
do
 echo "patch cherry picked = 11987$i"
 echo "fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/11987$i/1 && git cherry-pick FETCH_HEAD"
 git fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/11987$i/1 && git cherry-pick FETCH_HEAD
 $((i++))
done

echo " Applying first set of patches from 1180851 to 1180853 "
i=51
while [ $i != 54 ]
do
 echo "patch cherry picked = 11808$i"
 echo "fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/11808$i/3 && git cherry-pick FETCH_HEAD"
 git fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/11808$i/3 && git cherry-pick FETCH_HEAD
 $((i++))
done

echo " Applying first set of patches from 1219207 to 1219209 "
i=07
while [ $i != 10 ]
do
 echo "patch cherry picked = 12192$i"
 echo "fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/12192$i/1 && git cherry-pick FETCH_HEAD"
 git fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/12192$i/1 && git cherry-pick FETCH_HEAD
 $((i++))
done

echo " Applying first set of patches from 1219210 to 1219216 "
i=10
while [ $i != 17 ]
do
 echo "patch cherry picked = 12192$i"
 echo "fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/12192$i/2 && git cherry-pick FETCH_HEAD"
 git fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/12192$i/2 && git cherry-pick FETCH_HEAD
 $((i++))
done

echo " Applying first set of patches from 1219217 to 1219224"
i=17
while [ $i != 25 ]
do
 echo "patch cherry picked = 12192$i"
 git fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/$i/12192$i/1 && git cherry-pick FETCH_HEAD
 $((i++))
done

echo "patch cherry picked = 1219225"
git fetch https://chromium.googlesource.com/chromiumos/third_party/kernel refs/changes/25/1219225/2 && git cherry-pick FETCH_HEAD
```

Figure 3.1: Script to Automate the Porting

patches in which first I had to classify the patches of audio, then I had to classify the SOF and legacy audio patches. Each patch I had to check for classification. After classification, 150+ patches had to be ported from the master branch to Intel internal branch. Manually Cherry-picking each patch was not convenient. shell script has to be developed for this task. I had written a shell script which can cherry-pick all the patch and cleanly apply. Merge conflict had to be resolve manually. Figure 3.1 is the code snippet of the script. After successful porting, I had pushed to Intel internal branch. I had to build the kernel and test the basic audio functionality which is available

(3) ebuild changes for SOF Firmware & Topology File

Time Spent :- 3 Days

Summary :- In the SOF Development, SOF and SOF Topology file can be generated using the source code but we have to manually add those two files in the system which

	Subject	Status	Owner
Outgoing reviews			
☆	FROMGIT: ASOC: SOF: Remove always true condition in msgs		Shah, Smit
☆	FROMGIT: AOSP: SOF: Fix header guard typo		Shah, Smit
☆	FROMLIST: ASoC: Intel: common: add Geminilake Realtek+Maxim machine driver entry		Shah, Smit
☆	CHROMIUM: Config: Enable Realtek+Maxim Machine driver for GLK		Shah, Smit
☆	CHROMIUM: ASoC: SoF: Intel: Use standard Realtek+Maxim machine driver		Shah, Smit
☆	chromeos: add required config for SOF HDMI		Shah, Smit
☆	BACKPORT: ASoC: hdac_hdmi : Ensuring proper setting of output widget power state		Shah, Smit
☆	FROMGIT: ASoC: Intel: Fix build		Shah, Smit
☆	UPSTREAM: ALSA: hdac: Remove usage of struct hdac_ext_device and use ...		Shah, Smit
☆	UPSTREAM: ALSA: hdac: Remove usage of struct hdac_ext_bus and use hdac_bus ...		Shah, Smit
☆	UPSTREAM: ALSA: hdac: Remove usage of struct hdac_ext_driver, use hdac_driver...		Shah, Smit
☆	UPSTREAM: ALSA: hdac: ext: add wait for codec to respond after link reset		Shah, Smit
☆	UPSTREAM: ALSA: hda: split snd_hda_codec_new function		Shah, Smit
☆	UPSTREAM: ALSA: hdac: remove memory allocation from snd_hdac_ext_bus_device_init		Shah, Smit
☆	UPSTREAM: ALSA: hdac: add extended ops in the hdac_bus		Shah, Smit
☆	FROMGIT: ALSA: hda: move hda_codec.h to include/sound		Shah, Smit
☆	FROMGIT: ASoC: Intel: common: add table for HDA-based platforms		Shah, Smit
☆	FROMGIT: ASoC: Intel: Boards: Machine driver for SKL+ w/ HDAudio codecs		Shah, Smit
☆	FROMGIT: ASoC: Intel: Skylake: use HDAudio if ACPI enumeration fails		Shah, Smit
☆	FROMGIT: ASoC: Intel: Skylake: add HDA BE DAIs		Shah, Smit
☆	FROMGIT: ASoC: Intel: Skylake: use hda_bus instead of hdac_bus		Shah, Smit
☆	FROMGIT: ASoC: hdac_hda: add asoc extension for legacy HDA codec drivers		Shah, Smit
☆	FROMGIT: ASoC: SOF: debug: Use signed type for PM return value.		Shah, Smit
☆	UPSTREAM: ALSA: HDA: Fix several mismatch for register mask and value		Shah, Smit
☆	FROMGIT: ASoC: SOF: Define sof_create_platform_device for platform device ...		Shah, Smit
☆	FROMGIT: ASoC: SOF: refine and cleanup for request_firmware		Shah, Smit
☆	FROMGIT: ASoC: SOF: add hda_bus to struct sof_intel_hda_dev to handle hda ...		Shah, Smit
☆	FROMGIT: ASoC: SOF: hda-stream: add CORB/RIRB ringbuffers init for HDA.		Shah, Smit
☆	FROMGIT: ASoC: SOF: Add Kconfig item for SOF HDA.		Shah, Smit
☆	FROMGIT: ALSA: HDA: export process_unsol_events() which can be used without ...		Shah, Smit
☆	FROMGIT: ASoC: SOF: HDA: add hda codec probing APIs		Shah, Smit
☆	FROMGIT: ASoC: SOF: HDA: add hda bus initialization API		Shah, Smit
☆	FROMGIT: ASoC: SOF: HDA: introduce hdac_bus for probing and stream management		Shah, Smit
☆	FROMGIT: ASoC: SOF: topology: continue parsing when num_hw_configs is not 1		Shah, Smit
☆	FROMGIT: ASoC: SOF: PCM: do nothing for HDA dai_link fixup()		Shah, Smit
☆	FROMGIT: ASoC: SOF: hda-dai: add ops for hda link dais.		Shah, Smit
☆	BACKPORT: ALSA: hda: Copying sync power state helper to core		Shah, Smit
☆	FROMGIT: ASoC: Intel: bxt_da7219_max98357: enable HDMI for SOF		Shah, Smit
☆	FROMGIT: ASoC: SOF: Move xtensa oops/stack dump out of SOF core		Shah, Smit
☆	CHROMIUM: ASoC: SOF: Intel: hdac_hdmi_jack_port_init param modified.		Shah, Smit
☆	CHROMIUM: Config: Changes to enable the HDMI for SOF.		Shah, Smit
☆	CHROMIUM: ASoC: SOF: Intel: Enable HDMI Audio		Shah, Smit
☆	FROMGIT: ASoC: SOF: enable S2idle entry on SKL+ platforms		Shah, Smit
☆	CHROMIUM: ASoC: SOF: Remove Debug Flag		Shah, Smit

Figure 3.2: Ported Patch List

is not feasible. There are ebuild files in the Chromium OS source code which we can access and modify according to requirement. I want to make such change that this two files automatically be added in the given location in Chromium OS. First I need to find that ebuild file and then I had to understand how that ebuild works. After Knowing the structure I had to write a code, that can automatically add firmware and Topology file. Build the Chromium OS image and check the audio sanity. I had pushed the patch to Intel internal tree, all the developers and contributors of Chromium OS can take a benefit of it. This ebuild repository is limited to Intel and Google hence I can not show the code and patch.

(4) Buffer I/O error on dev loop7, Logical block 0, Async page

Time Spent :- 1.2 Week

Issue Detail :- This issue was reported on JIRA. It was not easy to reproduce. The procedure had been given to reproduce the issue. I had flashed the CPFE image, as per detail it was very sporadic. To reproduce the issue I had to reboot the system for multiple times, after doing a reboot for the 8th time, I was able to reproduce the issue.

Root Cause :- As per the message, It had to do something with the file system. I had done an analysis, only three possibilities for an issue.

First possibility was the hardware corruption, but it passes all the hardware test. The second possibility was the bad superblocks. The file system of Chrome OS made out of logical blocks within the superblock. I had done a test on the logical block but there was no error. The third possibility was related to buffer. I had printed the bufferhead address for multiple times, If it stops at the certain fix location all time, then there was something wrong with memory. All the time it was giving the different bufferhead location so buffer issue.

After more analysis, I had found the root cause, completion of the input and output operation Buffer will try to close, there were few async pages which some process was using and at the same time some other process will also try to access it, that's why it will not properly close and print this error message. The issue doesn't have any functionality impact.

Solution :- Unknown fix.

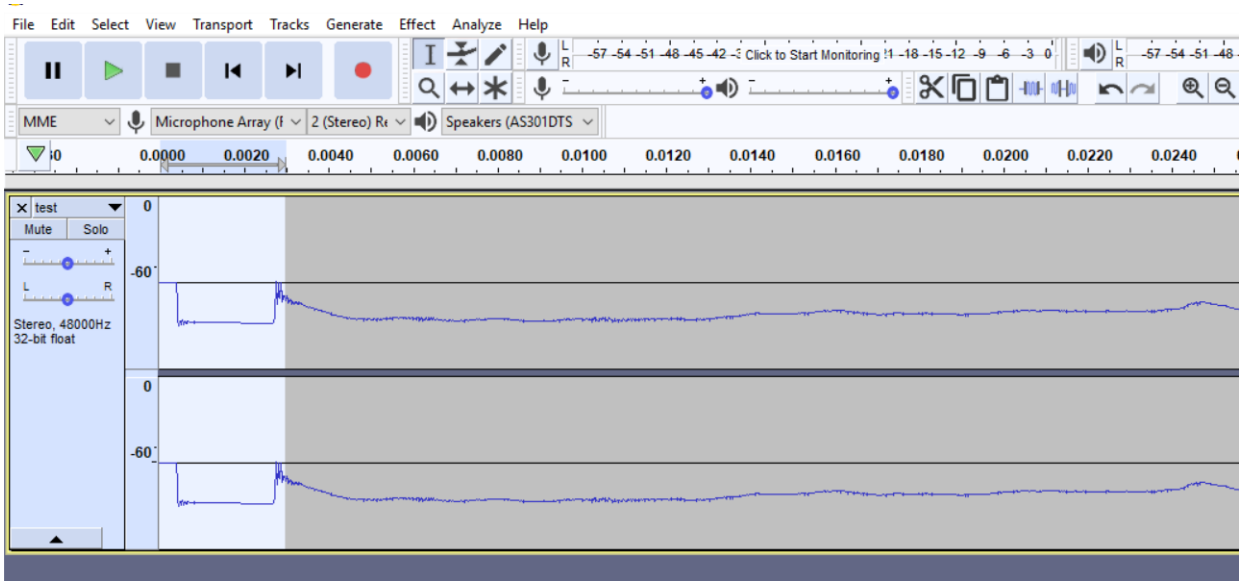

Figure 3.3: DMIC pop Noise Issue

(5) DMIC Loud pop noise at beginning of recording

Time Spent :- 2 Week

Issue Detail :- This issue was reported on JIRA. It was easy to reproduce, it can be seen on the multiple Intel Platform. According to the reporter when audio recording starts using the DMIC, one strong pop noise can be observed at the beginning of playback. To differentiate the pop noise with the outer noise, few additional tools required. Audacity is a free and open source tool in the Linux which can use for multiple audio related stuff. After Recording the audio using the DMIC, I had open that audio in the Audacity and enlarge the first few millisecond frequency waves. As per shown in figure 3.3, there was a short pop noise sound in the beginning.

Root Cause :- Third-party application can be a buggy if it produce a pop noise. In this issue, DMIC was creating an issue. In the DMIC recording process, first few milliseconds audio will be muted because of various system level process. for an example in the DMIC audio capturing process, starting 200ms will be muted with 80% volume using 100 quantization step process. This will create a smoother transition for audio muting but in the current case, it was not properly aligned. After providing data and logs, Firmware team increased the first 200ms mute to 230ms mute so, with a smoother transition, there will be no noise.

Solution :- Firmware changes has to be done for muting process. After the changes, issue get resolved.

(7) Validation Support For Multiple Issues.

Time Spent :- 2 Week

Summary :- Validation is as much important as development and integration. Developer submits the patches for issues but it needs to be validated. It is a responsibility of validator to perform a full validation of the issue, make sure that issue is resolved by the provided patch. In the validation, I have learned about the different types of validation techniques, post-merge validation, pre-merge validation. Daily many bugs are reported and collected from different sources, It is the responsibility of the validator to validate the issue and find out whether it is an actual bug or not. In an audio validation, I had performed different validation for the different issues, issues like for few media files audio was not routing to the headset, removal of HDMI Cable will cause audio to stop on system, suspend-resume test where we have to check for the regression in the suspend-resume functionality, validation I had to do it in a different environment with different recipe.

(8) Headset-audio, hot plug causes audio loss while streaming few media files

Time Spent :- 10 Days

Issue Detail :- This issue was reported on JIRA. Insertion of headset should be detected by the system. After the headset detection, audio should route to the newly attached headset. Removal of the headset should route the audio back to the speaker. In this issue headset was getting detected by the system but audio was not routing to the headset. To reproduce the issue. After system boot up, I had to insert the Headset. While playing the media files, Audio was not coming out of headset.

Root Cause :- In working condition, Headset should select the dual channel to route the audio, by checking the hardware parameter, Headset was selecting the 6 channel which is the HDMI, that was the root cause why audio was not routing to the headset. it was a firmware issue, we informed the firmware team about the issue.

Solution :- We received the firmware fix, Headset is now selecting the dual channel to route the audio. Tested the issue with fix, audio was routing to headset for those media files.

(9) Audio stops on extended monitor after device failing to enter suspend mode

Time Spent :- 10 Days

Issue Detail :- Insertion of the HDMI cable with the HDMI enabled monitor should be detected by the system. Extending or duplicating the screen can be used as a OS feature. In both mode, it should support audio to route to external monitor by default. In the suspend & resume test, audio firmware will get into the deep sleep state in the suspend mode. In the resume mode, Audio firmware should awake and get load. In ideal use case, while playing audio on external monitor, audio should keep playing after suspend & resume test. In this issue, Audio was getting stopped on extended monitor after suspend & resume test. To reproduce the issue, I had to connect the external display via HDMI cable. I had to play the YouTube video, audio should be routing to the external display. On the chromebook in shell, I had to start the suspend & resume test via suspend_stress_test script. Test was failed and audio stopped routing to the external display.

Root Cause :- Audio firmware was not going into the deep sleep state in suspend mode. In resume mode, firmware was not getting awake and load. It was a firmware issue. We informed the firmware team.

Solution :- Firmware fix solved the issue. After the fix, Audio is keep playing on external monitor after suspend & resume test.

(10) SOF binary infrastructure for Intel's Geminilake based Chromebook

Time Spent :- 15 Days.

Summary :- Firmware & Topology are the binaries required to enable the audio. Documentation is available on https://thesofproject.github.io for how to compile and generate the binaries. It was not reliable to copy those binary manually. There are two ways we can deliver these binaries to Google. Intel engineer will compile and generate the binaries and share it to Google via Partner tracker issue. Google will copy these binaries and host it on their local server to use it with Chrome OS. If any issue comes in binaries and after the fix, again Intel engineer needs to compile, generate and share it to Google

which is overhead and duplication of work. Another way is to create the ebuild file which would handle the binary generation dynamically. I had been given a task to create these ebuild file.

Chrome OS uses the Gentoo's portage as the package manager. ebuild file is the bash script which executes within special environment. It identify the specific software package and how to handle it. I had created the ebuild file to generate binary. Alsa-lib, Alsa-utils, crostool-ng, newlib xtensa are the different components required to compile and generate binaries. I also had to create ebuild files for different components which would also handle the runtime dependency. All the ebuild files I had created has the special flags. This special flag would help to only emerge single main ebuild file, that ebuild file had given the internal dependency which would be handled by portage. With the help of flag, all the ebuild would get emerge and generate binaries. if later any issues comes in firmware or Topology, and if we get fix, only git commit id needs to be updated in the ebuild file, everything else will be the same. With some modification Google can also use the same ebuild file for different platforms.

After binary generation, It needs to be tested to make sure the full audio functionality. Integrating the ebuild files in Chromium OS source code also needs to be test to check OS won't break while building. Build procedure to generate binaries gets changed so ebuild file couldn't make it to upstream.

(11) Running the IGT test suites on different environment

Time Spent :- 1 Week

Issue Detail :- IGT is the Intel GPU tools, that includes low level tools and test for development and testing of DRM drivers. IGT source code is open source and anyone can contribute. Validation team routinely runs the IGT test cases on the system to make sure, there is no failure.

After the validation, two of the IGT test case was failing on chrome system. It was easy to reproduce. Build the latest IGT test. Deploy it on target machine using the host machine. host machine will copy the test cases on target device and trigger. after the end of test, IGT script will return the result with full detail of logs. Plane flipping and Cropping test was failing. I was able to reproduce it on chrome system. while testing on the Ubuntu with same kernel version on same platform reference board. this two test

cases was passing.

Root Cause :- while checking the kernel and bisecting, few required patches were missing. bisecting the kernel further figured out the missing patches .

Solution :- After applying the missing patches in chrome kernel, test case were passing successfully. submitted the patches in Chrome OS kernel. Pre merge & post merge validation had been done to make sure no functionality break.

(12) Validation for full audio cycle.

Time Spent :- 4 Days

Summary :- Before making any audio release to Google, full validation cycle for audio is required. Validation process will ensures there is no issue in the current audio functionality, If any issue comes then we can fix it before making the release. with proper validation process, high quality product can be delivered.

In full audio validation cycle, all the test cases has been already defined. It covers the features like playback, recording, AV sync, suspend & resume stress. In the playback, functionality of speaker, headset, USB headset, external monitor using HDMI and DP cable gets validated. Use cases like hotplug of the device should detect by system, audio should route to the device after detection. unplug of the device should route the audio back. Audio should play without any noise and seamless playback is required to pass test. Playback from the different sources like web and local gets validate. In the Recording functionality, different recording scenarios will be covered. Recording in the DMIC, USB headset mic & Headset mic gets check. Recording should be seamless, smooth and without any noise.

In the A/V Sync, I had tested the media files from different sources like Web, Local, USB etc. Audio and video should be synced properly and audio should be clear and smooth to hear. I had tested the HDMI and DP monitor in both extended and duplicate monitor mode whether sync is proper or not. In the suspend & resume mode, I had checked the video playback, audio playback on speaker and external monitor. Audio should keep playing after suspend & resume test on device.

I had done the validation for different boad with different releases.

(13) Validation Support

Time Spent :- 7 Days

Summary :- Validation required the proper understanding on issue, expected behaviour and actual behavior.

I had done the pre merge & post merge validation on below listed issues. pre merge validation is when the fixes has not been merged in upstream chrome kernel. validation with local patch. Post merge validation is when the fixes had been merged in upstream and active in Chrome OS kernel.

- Unplug headset caused speaker mute.
- Sometimes GUI will not have Headset option listed.
- audio continues to play on removal of the USB which has source file.
- suspend & resume test with 100 iteration.
- Headset-audio, hot plug causes audio loss while streaming few media files.

(14) KGDB enabling on Chrome OS, Kernel Debugging

Time Spent :- 15 Days

summary :- Debugging can be very hard if you are not aware about right process. GNU debugger is one of the tool you can use to debug the code written in c, c++. Objective-c, Fortran etc but GDB can not use to debug the kernel. Kernel has special debugger named KGDB, We can set the software breakpoints, Hardware breakpoints, return register values etc. I had given task to find the BKM and try KGDB on Chrome OS. I had to start learning with minimum instructions given on Chromium OS developer documentation. following is the BKM. Prepared by me to use KGDB on Chrome OS.

- First step is to enable the KGDB flags in the kernel configuration. following are the configurations which needs to be enable.
  - KGDB: Kernel Debugger
    - KGDB: use KGDB over serial console
    - KGDB: allow debugging with traps in notifiers
    - KGDB kdb: Include kdb command functions to be enabled by default

    * KGDB: keyboard as input device
    * KGDB: continue after catastrophic errors

- Check whether kernel function symbols are enabled in kernel Makefile, if not then configure the Makefile with "kbuild_cflags+=-g".
- Build the kernel with USE="kgdb vtconsole" emerge-BOARD chrome-OS-kernel-VER (specify the board name and kernel version)
- Install the kernel on system and reboot.
- KGDB requires serial connection. since chromebooks doesn't have the serial port, servo board is needed to create serial connection. connect the servo board with host and target device.
- Use cros_sdk –no-ns-pid to enter the chroot in host machine. Start servo board with sudo servod -b octopus_npcx & (use specific board name instead of octopus_npcx), make sure there is no trace back warning and error.
- Use servo board command to check the port number, which servo board port is using for connection.
- Check the serial port number using the minicom console. check the dmesg in the target system for the serial port number. set the serial port number and baud rate in system's boot parameter and reboot,
- Freeze the kernel to it's current state by giving, echo g ¿ /proc/sysrq-trigger in target system.
- start GDB console on host machine, but do not stop the servo board running process. x86_64-cros-linux-gnu-gdb /build/octopus/usr/lib/debug/boot/vmlinux -ex target remote /dev/pts/5 (/dev/pts/5 can be different for others)
- GDB session should start without any error and warnings.
- Try setting hardware breakpoints and use stepping etc.

I had shared this BKM with other engineers in team. It has been verified, it is working. With my BKM debugging kernel using KGDB would be more easy. Engineers can

set the functional breakpoints and check the specific register values.

(15) Audio enablement on new Intel platform based Chromebook

Time Spent :- 15 Days

Summary :- After the successful release of SOF in Geminilake based chromebooks. SOF based audio solution has been decided for new Intel platform based Chromebook. Our Chrome OS team is responsible for the end delivery to Google. I had to start working on Audio enablement on new Intel platform based Chromebook. Initially we had to set up and use the Intel internal infrastructure which is based on 4.19 Chrome kernel. Topic branch has been created specifically for Audio patches where I need to port. Audio functionality has been split in the multiple releases.

In the 1st release, there was 190+ patches which I need to port. Released branch is for our reference, I can not take patches directly from released branch. ASoC patches are available in either audio maintainer's branch or in Patchwork which maintains mailing list patches. It is preferred to take patch from mainline upstream kernel if it is available there. All the patches should be in proper format. All upstream patch should have UPSTREAM tag, patch from maintainer's branch should be tagged as FOROMGIT, patchwork patch should be tagged as FROMLIST and local patches should be tagged as CHROMIUM. All the patches should have the location from where I pulled it and what is the original commit id. All the patches will be tagged with partner bug id and required test field, Bug Id is a partner tracker issue id, where Intel and Google will discuss about each release and issues.

Here below is the sample of final patch (Figure 3.4) which is merged in 4.19 Chrome kernel. Here in the figure 3.4, Patch is tagged as FROMGIT because it is from audio maintainer's tree. It has original commit id, location, Bug Id and Test field. I had started taking patches and apply cleanly. If any patch had conflict while applying then I had to understand the conflict and resolve it. To resolve the merge conflict either there will be some dependent patches which is required to pull before or few additional chromium patches is causing the conflict. To find the dependency I need to check the history on that file and see which patch is missing. If there is any missing patch then I have to apply it before. If there is any extra patch in the Chrome kernel then I have to manually do backport. Backport patch look like same as shown below in figure 3.5 with backport tag

```
FROMGIT: ALSA: memalloc: Add fall-through annotation

As a preparatory patch for the upcoming -Wimplicit-fallthrough
compiler checks, add the "fall through" annotation in
snd_dma_alloc_pages().  Note that this seems necessary to be put
exactly before the next label, so it's outside the ifdef block.

Signed-off-by: Takashi Iwai <tiwai@suse.de>
(cherry picked from commit 3c4cfa7bf6075be035cff3cac0986395f6fca32b
 https://github.com/broonie/asoc for-next)

BUG=b:123738217
TEST=Test Audio use cases for CML with full SOF patch series applied.

Change-Id: I591fae20a4dfa869a845348dd409608b6d2e4db1
Signed-off-by: smit shah <smit.shah@intel.com>
Signed-off-by: Ap, Kamal <kamal.ap@intel.corp-partner.google.com>
Reviewed-on: https://chromium-review.googlesource.com/1499065
Commit-Ready: Ben Zhang <benzh@chromium.org>
Tested-by: Ben Zhang <benzh@chromium.org>
Reviewed-by: Ben Zhang <benzh@chromium.org>
```

Figure 3.4: Patch from audio maintainer's tree

patch, I also need to add conflict detail.

After applying all the patch, I had to build the Chrome OS kernel and see whether is there any failure. Ported Kernel should have same functionality as released branch. Initially I had tested kernel with nocodec configuration where special codec will get loaded since board doesn't have codec on it. It should load the sof-nocodec configuration instead of HD Audio. To prevent the HD audio get loads, I had to blacklist the module in the /etc/modprobe.d/ location.

I had handled two audio release. All the patches I had ported initially in Intel internal infrastructure got merge and later reused to port in 4.19 chrome kernel. All the patches are available in Chrome 4.19 kernel. Before pushing patches, I had to do the sanity testing and check what is working and what is not working. In the sanity test, I had to check various Playback, Recording etc.

(16) Multiple audio autotests were failing with error

Time Spent :- 1 Week

Issue Detail :- Autotests are designed in such way that it will require less human interaction and it will test the functionality and give back the results. To get the autotest, While building the packages for Chromium OS, special flags needs to give. Autotest will

```
BACKPORT: FROMGIT: ASoC: Intel: Skylake: Add more platform granularity

The current SKYLAKE kconfig is a all-you-can-eat selection that will
support all known plaforms. This is however not necessarily a good
thing: most platforms for SKL and KBL don't support the DSP, but a
number of CNL/WHL ones do. Selecting this driver in all cases isn't
really smart and will require users to muck with blacklists.

Partition the configs to allow distributions to select on which
platform this driver is used. Keep the existing SND_SOC_INTEL_SKYLAKE
config to select everything for backwards compatibility. This patch does
not provide new functionality, only finer-grained choices in supported
platforms.

Signed-off-by: Pierre-Louis Bossart <pierre-louis.bossart@linux.intel.com>
Signed-off-by: Mark Brown <broonie@kernel.org>
(cherry picked from commit 35bc99aaa1a3af23cf78b6b56f14230b5da3993b
 https://github.com/broonie/asoc for-next)

Conflicts:
   sound/soc/intel/boards/Kconfig

BUG=b:123738217
TEST=Test Audio use cases for CML with full SOF patch series applied.

Change-Id: I7f7be3aec9af0db687a3ebffd190c73d00bd4b44
Signed-off-by: smit shah <smit.shah@intel.com>
Signed-off-by: Ap, Kamal <kamal.ap@intel.corp-partner.google.com>
Reviewed-on: https://chromium-review.googlesource.com/1499098
Commit-Ready: Ben Zhang <benzh@chromium.org>
Tested-by: Ben Zhang <benzh@chromium.org>
Reviewed-by: Ben Zhang <benzh@chromium.org>
```

Figure 3.5: Backport patch format

be triggered from host machine. Host machine will copy the tests in target machine and script will trigger the autotest script it on target machine.

In this issue, with loopback dongle recently added few audio test cases were failing. As per the reporter, connecting the loopback dongle, and triggering the tests, it should pass but it was failing. Loopback dongle is not available in the market. Google released the circuit diagram of loopback dongle that we had done the rework by Lab team. To reproduce the issue, I had to insert the loopback dongle in the headset jack and trigger the autotest from the host machine. All the five test cases were failing which is mentioned below.

- audio_AlsaLoopback
- audio_CRASFormatConversion
- audio_CrasLoopback
- audio_SeekAudioFeedback
- desktopui_AudioFeedback

Root Cause :- I had a suspect that it was a dongle issue. For the confirmation, I took the the different Intel platform based chromebook and ran the test on them with same dongle. On those systems also test were failing. We had ordered a new dongle and IT team has done the rework as per Google. With the new dongle out of 5 only 3 autotest were failing. out of 3, one I suspect because of the proxy issue in the Intel network. After resolving the proxy issue, audio_CrasLoopback was also passing. Root cause was the old loopback dongle.
Solution :- With the new Loopback dongle, out of 5 autotest, only 2 are failing now. Those two autotest which was failing are using the Chrome UI component, For Google also this two autotest are failing. Intel and Google both are working to fix the issue.

(17) SOF and SOF Topology infrastructure for new Intel platform based Chromebook

Time Spent :- 20 Days.
Summary :- As per mentioned in the issue & task 10, After the build procedure gets

changed to generate the SOF Topology & Firmware binaries. I had to reuse the old ebuild files and modify according to the new build procedure. After reading the documentation to generate binaries, two more new components has been introduced. In the old way, Alsa-lib, Alsa-utils, Crosstool-ng, new-lib xtensa, firmware source code and Topology source code components were there. In the new build procedure, xtensa-overlays has been introduced, firmware and topology source code had been merged in the one single directory. Now it requires to use the cmake version above 3.10. xtensa-overlays has the Intel platform specific configuration files requires to generate the binaries. I had modified the older ebuild files to use it with new platform. I had to create new ebuild files which will accommodate the new changes. I had created the ebuild file for the xtensa-overlays, It was also integrated in the other ebuild file. I had face the several challenges with new build procedure, because new source code requires to use the cmake version 3.10 and above. I can not use the chroot cmake.

Chroot is the special environment for chromium os source code maintained by Google. I can not upgrade the cmake version because other ebuild file in the chromium OS may get affect. To solve the issue and to give the demo to Google, I had created the new ebuild file for the cmake, which will download and install the cmake in specific location. I had tricked my SOF Topolog & Firmware ebuid file to not use the cmake from the chroot but take it from the specific location where my cmake installation is present. Now binaries can be generate using my ebuild file. Integration into the OS is on progress. Once all the ebuild file can emerge using the chromium os build_image. I can present it to Google to use Intel's solution for the binaries.

(18) New audio release for Intel's Geminilake based chromebook

Time Spent :- 20 Days.

Summary :- In the last audio release to Google for Geminilake platform, It has several issue. Patches were not upstream worthy and can create issue at long time if merged it into the Chrome 4.14 kernel. To resolve the issue, Intel and Google has decided to revert the last audio release. New v4 SOF patches are already present in the mailing list which will reduce the significant number of SOF patches, It was strong recommendation to use the V4 SOF patches. To use the new release and take the V4 SOF patches, I had to reverted 160+ around patches in the Chrome 4.14 kernel. In the revert process, I had to

find the revert chain or else it would not cleanly revert and conflict. With the help of Google and Intel engineer, All the 160 patches from old release had been reverted.

New V4 SOF patches, requires additional dependent ASoC patches. As per the release I took all the required ASoC patches and backported in the Chrome 4.14 kernel. Same as the mentioned in task & Issue 15. I need to give proper tag, new bug id and Test field. I need to specify the location of the patch. If any patch conflict while applying then I need to find the dependent missing patches or backport them manually by resolving the merge conflict. I need to make sure, after resolving the conflict manually, It should not break any functionality. I had backported every V4 SOF patch and required ASoC patches. While doing the sanity testing, Playback and recording was working but HDMI playback was not working. First we suspect that it is Firmware issue. After further checking, release was based on the Linux 4.14 kernel because of that it was not having the required i915 patches required for HDMI playback.

By the time V5 SOF patches are already landed in the mailing list and Intel strongly recommended to use the newer version of SOF patches. To resolve the HDMI playback issue, new release had been made using chrome 4.14 kernel. In this release, I had to revert the same older release patches using the revert chain. Again I had to take the required ASoC patches with all the proper format and tags. This time it was easy as most of the ASoC patches I can reuse from my older backport. I had done all the backporting with the V5 SOF patches.

After communicating with Google about new release. Google wants release in specific order to merge it easily in the 4.14 kernel. earlier it was revert chain + ASoC patches (Upstream + Fromgit) + SOF but now Google wants it in ASoC (upsream + fromgit) + Revert chain + SOF patches. I had arranged the patches in the correct order. After that Google wants all the Upstream patches in the upstream order so I need to resolve the order issue. After the sanity test, Allyesconfig is failing with my patches. fix is in progress.

I had created two patches and submitted to Google. One patch will normalize the kernel configuration and second patch will enable the required configuration to enable the SOF audio on chrome 4.14 kernel. Patches has been pushed to Google and it is in merging process.

Chapter 4

Conclusion

4.0.1 Conclusion

Next generation audio system using Sound Open Firmware has been introduced to eliminate the issue with closed source DSP firmware. With the help of SOF now contributors and developers can easily integrate their pre-written algorithms. Developers can debug the firmware issue more easily. Open-source nature will help SOF for more contribution from community.

Bibliography

[1] G. LLC, "Chromium os architecture." https://www.chromium.org/chromium-os/chromiumos-design-docs/software-architecture.

[2] G. LLC, "Chromium os." http://www.chromium.org/chromium-os.

[3] L. foundation, "The sof project." https://thesofproject.github.io/latest/introduction/index.html.