VoIP DSP Functionality in Software on Linux Based Power Pc Platform

Hemant Agrawal, *Member, IACSIT*

***Abstract*—These network devices in small and medium business and home market are incomplete without supporting voice over IP. Traditionally dedicated DSP based solutions are used for processing the packetized voice with Quality of Service support. However it is not cost-effective for the low-end devices to have a DSP on board. One of the challenges in providing a VoIP solution is extracting the same level of performance from a Linux based PowerPC that a DSP would offer. This paper covers the challenges in implementing VoIP framework on Linux. This results in a "software-DSP" implementation on Freescale's Power PC platforms.**

***Index Terms*—VoIP, DSP, PowerPC, speech coding, real time patch.**

I. Introduction

Voice over Internet Protocol (VoIP) is a technology for delivery of telephony speech, facsimile and data signals over existing data networks. VoIP is also sometimes referred as Packet telephony and Internet telephony. A VoIP Gateway is required to convert the media from one network type to another.

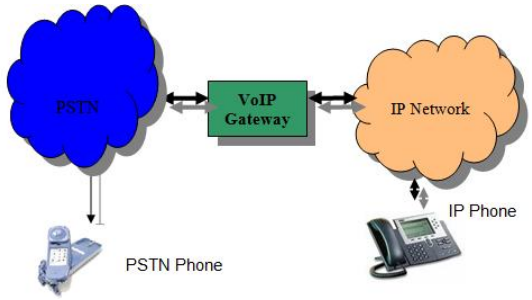

Fig. 1. VoIP gateway

VoIP systems are implemented in three layers as follows:

A. Signaling Control Layer

Signaling control layer controls the state of connection between VoIP endpoints. There are standard protocols to achieve the purpose. E.g. SIP, H.323, and MGCP etc.

B. Media Processing Layer

Media Processing Layer is responsible for conversion between two different formats and handling of various media events. A typical case is TDM/PSTN to IP packets conversion and vice-versa. Some of the voice component of this layer are G.711, G.723.1, G.729 A/B; voice activity detect; packet loss concealment; acoustic and line echo cancellers; DTMF detection and generation; adaptive jitter buffer; RTP/RTCP/UDP/IP etc.

Manuscript received January 6, 2012; revised April 21, 2012. This work was supported by the Freescale Semiconductor, India.

Hemant Agrawal is with the Freescale Semiconductor, Plot-18, Sector 16A, Noida, UP-201301, India (e-mail: hemant@freescale.com).

C. Media Processing Control Layer

It provides interface between Signaling control layer and the Media Processing Layer. In a typical use case, it interfaces with Signaling layer via function calls and it interacts with VoIP framework via network communication.

Traditionally VoIP end-point typically has an architecture based on a combination of a DSP and a general purpose CPU. The computational intensive work of media processing is performed by the DSP, and the general purpose CPU performs the system management, signaling, call control and network data processing. Freescale's voice solutions are being offered as a two processors solutions. These two processors can be part of a single SoC offering (e.g Freescale's MSC7120). However, it is not cost-effective for low-end devices to have an on-board DSP. Freescale's low-end QorIQ devices are capable of providing the low channel voice solution (2-8) using a PowerPC processor without using the DSP. This results in a software solution implementing DSP speech codecs, voice algorithms, echo cancellers, telephony signaling, IP call control, network packet processing and system management on a single core running an embedded Linux operating system. This solution can also be based on multi-core processors (Freescale P1020)[1], where one core can be dedicated for media processing to achieve higher channel density.

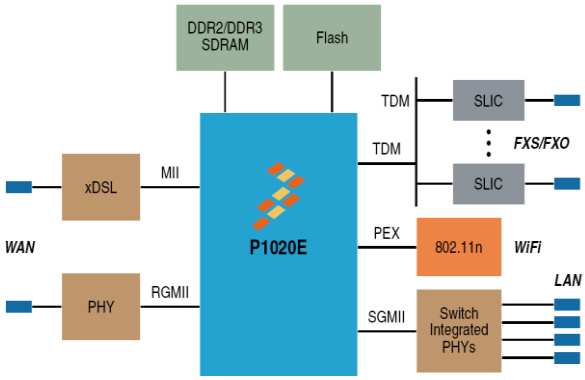

Fig. 2. Freescale P1020E with inbuilt TDM support

The advantages of a DSP-free architecture for VoIP-enabled end-points are very appealing; however, it creates several development and software architecture challenges. One of the challenges in providing a VoIP solution is extracting the same level of performance from a PowerPC that a DSP would offer for the underlying voice components, which are traditionally suited to the DSP type of architecture. This challenge become more complex when the control and media processing functions are merged onto a single device, care must be taken to control the overall system priorities to address the real-time nature of the voice

processing. All elements of the Power PC architecture from best-use of the pipeline to elimination of unnecessary instructions and parallelism must be considered if a complex task such as VoIP is to run in real time.

Another challenge is that Linux, being a general-purpose OS, is not designed to support applications having real-time requirements while Voice processing requires soft real-time guarantees. Any scheduling delay and missed dead-lines can deteriorate the perceived quality and user experience.

In this paper we will discuss how to mitigate the challenges in implementing a software-DSP VoIP framework on Linux ased Freescale Power PC platforms. We will explore how Linux and the PowerPC architecture can be leveraged to meet the VoIP processing requirements.

II. Procedure for Paper Submission Building the software DSP based Voice Processing Solution

Since both network and TDM (Time Division Multiplexing) drivers are available in Linux Kernel Space, developing the VoIP Framework in kernel space is an obvious choice to avoid any buffer copy and limit scheduling issues. Once the PCM data is received on the TDM bus, the codec encrypts it and places it in a network buffer after applying various voice algorithms for event detection and quality improvements. Once the packet is ready, after packetizing it with RTP, the packet can be transmitted without making a copy. In the reverse path, use of a carefully designed adaptive jitter buffer addresses the key voice quality issues that packetize the voice experience due to a number of factors such as delay and jitter in the network. Another key factor is the delay in processing voice due to other activities by the CPU, etc. These issues are addressed by the careful design and implementation of Packet Loss Compensation (PLC), Comfort Noise Generation (CNG), careful design and use of Jitter Buffers, Echo Cancellation (ECAN), and OS modifications to handle Voice in real time [2].

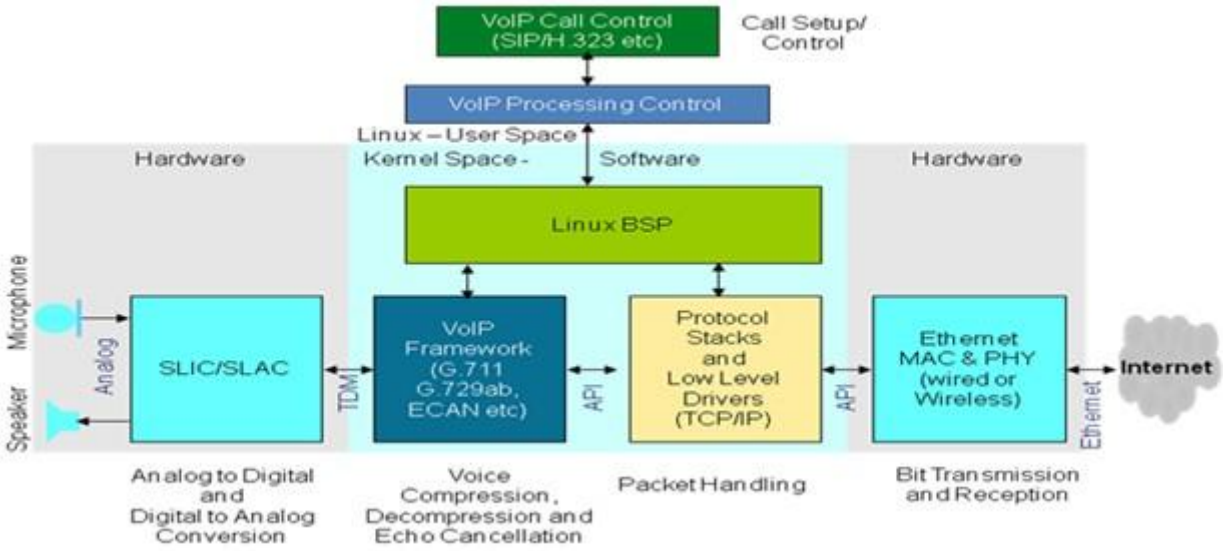

Fig. 3. Linux based solution

A. DSP Speech Codecs and Algorithms on Linux

The VoIP framework has several components such as voice codecs (G.711, G.729 etc), echo canceller, RFC2833, tone detection etc. These components are computation intensive and are traditionally suited for a DSP (Digital Signal Processor).

PowerPC Assembly instructions do not have floating point support, so there is a Fixed Point implementation for these components. Initial results of running the raw "C" model of commonly used voice codec, G.729AB, showed that it takes close to 200MHz just to process one voice channel data. This made it very clear to optimize these voice components for the platform.

The following methods have been adopted for profiling and optimization of voice algorithms.

1) **Profiling –** First step was to measure the raw performance of the C-Model. Function wise profiling helped in determining which section of the code is taking maximum cycles.
2) **Optimization of Basic Op functions -** Voice algorithms generally use a lot of basic op functions such as **L_add, L_sub, L_mult**. These functions are fixed point arithmetic functions. Since the number of calls of these functions is generally very large, they tend to use a lot of CPU cycles. These functions were the first target for optimization. Highly optimized Basic OP library using PowerPC assembly was developed. Ideally, the fixed point algorithm should be written so as to insure that there is no overflow. However, a few cases can occur where the chance of overflow still remains. The code was analysed for overflow conditions - Visual Analysis and executing it with hundreds of test vectors. Safe conditions were changed to a simple set of macro based basic Ops.
3) **Basic C optimization –** Manual optimizations include loop unrolling, merging multiple loops, minimizing load/stores, etc.
4) **Assembly level optimization** - for CPU intensive areas of code.

Linux has a default stack usage limitation (4K default) for

better performance. Our DSP based VoIP framework and Codec processing was full of nested functions, easily overshooting 4K or 8K stack sizes. We reorganized the code to avoid too much nesting.

B. Optimization Results:

After the first round of optimization, the following results were achieved.

TABLE I: VOICE COMPONENT OPTIMIZATION RESULTS

Module	Maximum MCPS on MPC8315			
		Fixed point raw C-code	First Optimized Code	Target for Prototype
G.711	*Encoder*	5.2	1	1
	Decoder	1.86	1	1
Echo Canceller		187.35	27	30
RFC2833-DTM Detection	*Encoder*	30	8	10
	Decoder	2.71	1.5	2
Tone Generation		8	1.5	3
G.729	*Encoder*	163.75	100*	40
	Decoder	41.73	30*	10

* Current status

Note: MPC 8315 Core (e300) clock frequency is 400MHz. Frame size is 10ms @ 8 kHz sampling frequency (MCPS = Million Cycles Per Second).

C. System Scheduling

The Linux kernel has many processes applications running. An application/thread can always be interrupted by higher priority ISRs (hard and soft). Spinlock (depends on number of cpus) can make you wait. The whole nature of Linux as an OS is not deterministic. On the other hand, voice processing needs to meet the hard TDM timelines. TDM is synchronous processing, which works by using TDM interface tick.

In a typical home router system, there are three categories of applications:

1) VoIP – CPU intensive - Real-time with latency requirement of 5-10ms.
2) **Internet** Packet Forwarding – Lower than Voice – 1ms processing requirement at higher rates.
3) Applications – such as UI – 500ms class – it should not be CPU starved (usage are less than 5%).

We initially implemented the voice processing in the context of a softIRQ triggered by TDM interrupt. On low network load conditions, the system was able to support good quality of voice. However as the network traffic increased, the voice processing began to starve. Increase in packet loss also caused a severe impact on voice quality.

We enabled the RT_PREEMPT patch [3], [4] in Linux kernel, which provided:

- Almost full-preemption
- Threaded interrupts - ability to priotize interrupts like processes
- Lower risk of over/underruns

III. TEST RESULTS

The following chart shows the test result of PESQ scores measured by ABACUS for MPC8315 based solution.

1) 2-way G.729ab call
2) Shows PESQ (ITU-T Perceptural Quality) scores for 15 1 minute calls. Higher is better.
3) Comparisons are made against two industry-standard products from Cisco and UTStarComm.

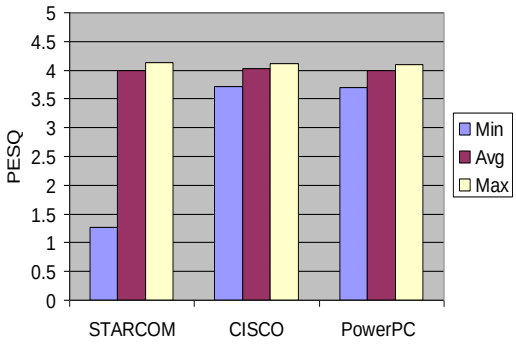

Fig. 4. Voice quality comparision

4) This test evaluates the overall quality of the decode path.
5) Small differences in PESQ are not perceptible.
6) PowerPC is running with IPv4 forwarding at 70% line rate.

On further evaluation, the behavior of the system with increasing the network traffic, it was observed that after 75% line rate, the call does not get established due to packet loss.

TABLE II: RESULTS G.729AB VOICE QUALITY COMPARISON

Packet Size (in Bytes)	64 Byte Packet Size			1500 Byte Packets		
Line Rate (in %)	64-Byte Min	64-Byte Avg	64-Byte Max	1500 Byte Min	1500 Byte Avg	1500 Byte Max
60	3.19	3.991	4.147			
70	3.713	4.038	4.121	3.218	4.016	4.124
72	3.563	4.012	4.127			
73	2.638	4.021	4.127			
74	2.929	4.013	4.137			
75	1.824	3.715	4.148			
80	0	0	0	3.106	4.01	4.145
90	0	0	0	2.963	3.686	4.125
100	0	0	0	3.132	3.725	4.12

Going forward, QoS for VoIP packet can be enabled to give priority treatment for the VoIP traffic.

IV. CONCLUSION

In an actual implementation on Freescale's MPC8315 processor, the solution was able to support two simultaneous voice channel support with just 90 MCPS usages for a 2-way call (G.711). The channel function definition included G.711 voice codecs, DTMF Detect/Generate/Relay, Echo Canceller with a 16 ms tail, Tone handling, RTP/RTCP, Adaptive Jitter Buffer Manager and Call Control. Considering the 400MHz processor, the results shows that 2 calls with G.729ab can easily be supported using less than 150MHz total once the G.729ab optimization exercise is over. Further optimization in voice components and Linux can be undertaken to achieve the same in less than 100 MHz. With the QorIQ series processor running at 800 -1200 MHz and supporting the improved e500 core, this processing power requirement for

supporting the voice channel will decrease; resulting in better channel density and scope to add more functionality.

REFERENCES

[1] Freescale QorIQ P1020. [Online]. Available: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=P1020.

[2] M. Felice, "Building DSP-Free VoIP End-Points," *Information Quarterly*, vol. 3, no. 4, pp. 41, 2004.

[3] X. Wang, "Solving Real-World Real-Time Scheduling Problems With RT_PREEMPT and Deadline-Based Scheduler," *Embedded Linux Conference* 2011.

[4] Real Time Preemption in Linux. [Online]. Available:http://elinux.org/Realtime_Preemption.

Hemant Agrawal is a software architect with Freescale Semiconductor, Noida, India. He has over 13 years of industry experience in design and development of networking and telecommunication systems and applications. His Primary focus and expertise is in Networking acceleration software, IPSEC, voice over IP, SIP, H.323, SS7, ISDN Q.931, MGCP, MEGACO protocols. He was part of IETF ś SIP-H.323 interworking standard development team. Hemant holds a B.Tech. In electrical engineering from institute of technology, banaras hindu university (IT-BHU), Varanasi, India.