A MEMORY-EFFICIENT ALGORITHM FOR NETWORK ECHO CANCELLATION IN VOIP SYSTEMS

Anil Ubale

Intel Corporation
350 E. Plumeria Drive
San Jose, CA 95134, USA

ABSTRACT

In recent years, considerable interest has been focused on voice-over-IP (VoIP) systems. VoIP media gateways implement voice and packet processing for hundreds of channels. One of the limiting factors in such high channel-density voice processing applications is the memory requirement. A substantial amount of memory is used to store the coefficients of network echo canceller. This paper describes a new algorithm that reduces the memory requirements for echo cancellation by a factor of 2 to 4. When the new memory reduction algorithm is combined with standard NLMS, it also provides faster convergence rate, and better echo path tracking than the standard NLMS algorithm.

1. INTRODUCTION AND BACKGROUND

In recent years, communication networks and services based on the Internet protocol (IP) have grown by at a tremendous rate. The continued rapid advance of the internet has made it necessary to the convergence of the public switched telephone network (PSTN) and the internet. A considerable interest has been focused on Voice-over-Packet (VoP), or Voice-over-IP (VoIP) systems. The transition of media signals from circuit-switched PSTN and packet-switched IP network is accomplished in a VoIP system is accomplished in a media gateway. A media gateway includes several signal processing components that operate to convert voice signals into a stream of packets that are sent over a packet-switched network such as the Internet and convert the packets received at the destination back to voice signals. A simplified logical flow diagram of a media gateway is shown in Figure 1 [1]. The voice processing component of the media gateway system transforms time-division multiplexed (TDM) voice signals from the PSTN side to IP packets and vice-versa. The voice processing component, usually implemented on a media processing card has to perform telephony and voice processing functions on hundreds of TDM (DS0) [2] channels.

Supporting high channel density is a significant challenge. In Intel® media gateway solution the voice processing component is implemented on a digital signal

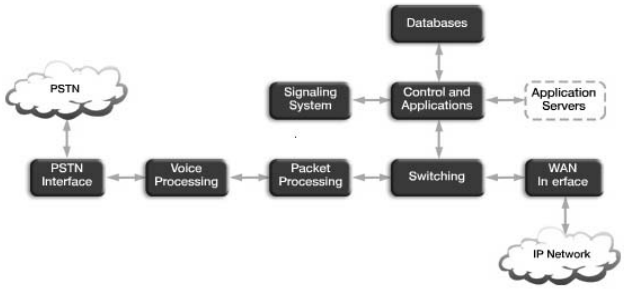

Figure 1. VoIP Media Gateway system logical flow.

processor IXS1000 [1]. As shown in Figure 2, the IXS1000 incorporates multiple signal processing cores, a control processing core, and a high-speed internal bus, and a memory movement engine. The signal processing cores have tightly-coupled RISC and DSP resources, and local instruction and data memories. The signal processing cores are further assisted by global memory on the chip.

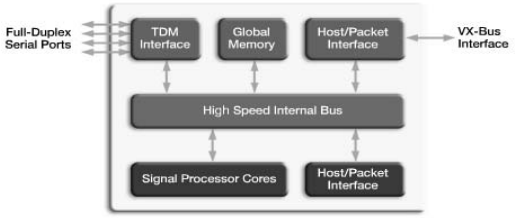

Figure 2. Intel IXS1000 Signal Processor Block Diagram.

Factors affecting the number of channels supported in a voice processing component of a media gateway system are the MHz and memory requirements for each channel. In IXS1000 the on-chip global memory SRAM stores the firmware including, program, tables, and state information data for all channels. Among various voice processing

tasks is network echo cancellation. The network echo canceller requires a large amount of state memory per channel to be stored in the global memory. This is especially true when echo cancellers for echo paths with long impulse responses (>= 64 ms) are desired [3].

Generally, the voice processing per channel is performed every frame. A frame is a time interval used to generate a voice packet to be transferred to the IP network and vice-versa. The frame size is typically dictated by the voice compression used. For sample-based voice compression methods the frame size can be arbitrarily chosen. In this paper, we assume a frame size of 10 ms or 80 samples. The state memory required by the echo canceller to perform its corresponding function is transferred via direct memory access (DMA) mechanisms into and out of the signal processing cores from and to the global memory for every frame. Typically, a DSP device can be configured to perform voice and packet processing for multiple VoIP channels and each VoIP channel can have its own persistent state memory storage in the global memory. Accordingly, the number of VoIP channels that can be processed depends, among other factors, on the state memory requirements of the echo canceller. Thus, higher channel densities can be achieved if the state memory requirements of the echo canceller can be reduced without significant quality impairment of voice signals.

A simplified block diagram of a network echo canceller is shown in Figure 3. In general, a substantial portion of the state memory requirements of an echo canceller is allocated for storing coefficients of the adaptive filter $h(n)$. For example, for a 128 ms echo canceller the filter length is 1024, and the state memory required to store the coefficients with 16-bit precision is 2 Kbytes. In this paper, we describe a new method to reduce the memory requirements for storing the coefficients of the echo canceller by a factor of 2 to 4. We also show that the new method provides faster convergence rate than the traditional normalized least mean square (NLMS) [4] algorithm.

2. A MEMORY EFFICIENT ECHO CANCELLATION ALGORITHM

As a frame of voice samples for a particular channel is scheduled for echo cancellation, the echo canceller coefficients corresponding to that channel are first transferred from the global memory to local memory. An adaptation algorithm that minimizes the echo, e.g,. NLMS, is run every sample. Thus the coefficients get updated every sample in the local memory. At the end of the frame, the coefficients are transferred from local memory to global memory, and the processing for next channel can the data compression of the echo canceller coefficients is performed. Thus the memory requirements for a 1024-tap

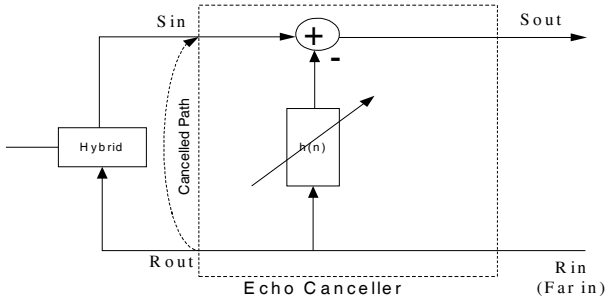

Figure 3. Simplified block diagram of a network echo canceller.

echo canceller coefficients are 2 Kbytes in local memory but significantly lower when stored in the global memory. When the same channel's next frame is scheduled to be processed by the signal processing cores, decompression of the data is performed, so that the filter coefficients are recovered in 16-bit precision. The trick is to minimize the loss of precision in the most important taps of the echo canceller. The most important taps of an echo canceller are the ones with the highest magnitude. Therefore, our new method aims to retain full precision for the highest coefficients.

Let us think of the coefficient array as two arrays -- a sign-bit array and a magnitude array. The magnitude array requires 15 bits per coefficient. Let us assume that a 50% reduction in memory requirements is desired. Thus for a 1024-tap echo canceller we will have a bit-quota of 8192 bits per frame or 8 bits per coefficient. Now consider the magnitude array as a two-dimensional array, row numbers are taps or coefficients, and column numbers are bits. Thus, rows are coefficients 0 to 1023 from top to bottom, and columns are bits 14 to 0 (MSB to LSB) from left to right. This is a 1024 X 15 array. Note that, we could get some savings just by throwing away the 0 bits in the left-side of this array and without incurring any loss! Thus we code the left-most non-zero MSB position. That is, we traverse from left top of the array to right bottom of the array in a zig-zag fashion. Down to bottom, then up to top and right, then down again, till we hit first 1 bit! The position of this bit is coded using 4 + $log2$(TapLength) bits, i.e., 15 bits in our example. The 4 bits are needed to specify the column position and $log2$(TapLength) to specify the column position. Note that, we just found the highest-coefficient, we immediately pack this coefficient with full precision. The number of bits required is given by the column position coded with 4 bits already. Next we pack the sign bit corresponding to this coefficient using 1 bit. We also pack 4 coefficients adjacent to the highest coefficient with the same precision as that of highest coefficient. Although this might waste a few bits to specify the precision, it saves bits required to specify the row position of the coefficients. This improvement is motivated by the fact that the coefficients next to the echo

path peak are also high. Once we code these coefficients, we zero-out its bits in the magnitude array, i.e., this row is all zeros now. We continue traversing in the same zig-zag manner and as soon as we hit a 1, we first code the position of the new 1. Immediately we also code this coefficient with full precision as required. The position is coded relative to the previous position. In essence, we are choosing the strongest coefficients and coding them and their positions. Also note that we are coding them with just enough precision and not more and not less. We continue this process till we either run out of bits (bit-quota) or reach the right bottom of our two-dimension array.

The algorithm steps are specified below:

1. n, coefficient index = 0. k, = bit position = 14.
2. Is k-th bit of $|h(n)| = 1$?
3. If no jump to step 6.
4. Encode n differentially relative to old n (=0 initially). Encode $|h(n)|$ with k bits and sign of $h(n)$ with 1 bit. Old $n \leftarrow n$. Set $h(n) = 0$.
5. Reduce bit quota by the bits used in step 4.
6. If bit quota is zero or less, STOP.
7. $n \leftarrow n+1$.
8. Is n = Tap length?
9. If no jump to step 2.
10. $k \leftarrow k$-1.
11. Is k = -1?
12. If no jump to step 2.
13. STOP.

Note that, this method zeros out the spurious coefficients, i.e., coefficients that are small (logarithmically). Therefore in case of sparse and quasi-sparse (dispersion is limited to few milliseconds) the method increases the convergence rate. In other words, every frame the small or near-zero coefficients are adjusted back to zero, thus allowing a faster adjustment of larger coefficients. This is similar in principle to the methods where the larger coefficients are given larger scale factors in order to improve convergence [5],[6],[7].

The savings in memory requirement, and faster convergence rate, however come at the expense of increased complexity. Every frame (typically 10 ms or 80 samples) the compression and decompression algorithm is executed adding to the MHz requirements. The algorithm presents a trade-off between memory die-cost savings vs. MHz increase. The algorithm can also be implemented in silicon, in which case the trade-off is between the memory and compression/decompression engine die-costs.

3. SIMULATION RESULTS

We compared the results of out new memory efficient algorithm with the standard NLMS algorithm. Note that, the coefficient compression/decompression can be combined with other LMS and affine-projection type algorithms as well.

The objective of these simulations is to compare the NLMS and the new memory efficient NLMS algorithms in the context of network echo canceller application. For our simulations we use two different echo paths as shown by their impulse responses in Figure 4. The sparse impulse response is the echo path model 1 from reference [3] and represents a single reflection echo. The dispersive impulse response represents multiple reflections and a dispersion of approximately 9 ms. This is echo path model 7 from reference [3]. The experiments use composite source signal (CSS) as defined in [3] and male speech signal as far in excitation input signals. The sampling rate is 8 kHz. Both the echo canceller tap length and echo path length are 1024 taps.

The compression and decompression of the coefficients is performed every frame, where the frame length is 80 samples. We compare results when the new algorithm provides memory reductions by a factor of 2 and 4.

Figures 6, 7, 8, and 9 compare the misalignment given by, $\|\mathbf{h} - \mathbf{h}'\|/\|\mathbf{h}\|$ for the NLMS and the new algorithm. We can see that the new memory efficient algorithms also provide faster convergence rate compared to standard NLMS algorithm. The convergence rate improvement is even better at higher memory reductions. However, using very high memory reductions may lead to slower convergence rate or divergence for highly dispersive echo paths. The improvement in convergence speed is consistent across both CSS and speech input signals.

We also confirmed that the algorithm has desirable echo path tracking properties and faster convergence rate when echo path changes as shown in Figure 10.

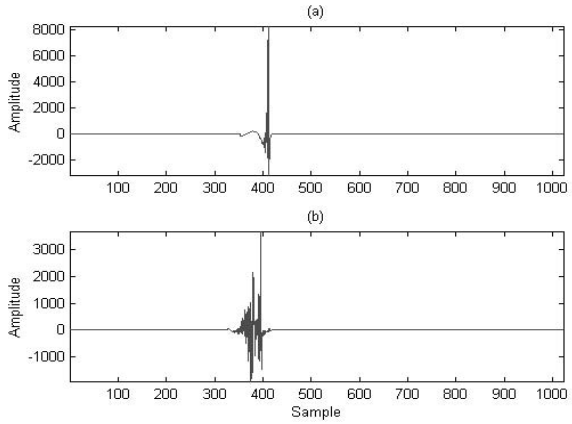

Figure 4. Two impulse responses used in the simulations (a) single reflection sparse, (b) multiple reflections dispersive.

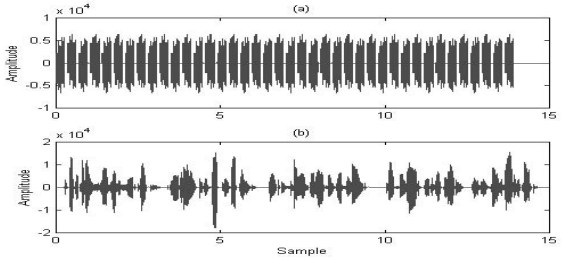

Figure 5 Input far in signals used in the simulations (a) Composite source signal, and (b) Speech signal.

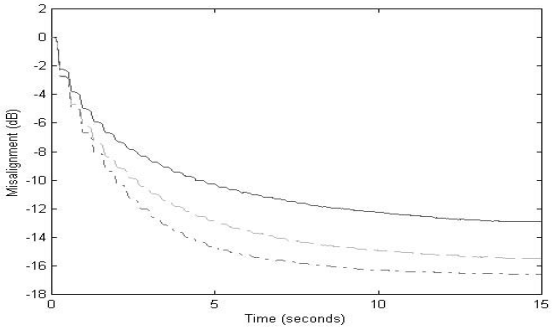

Figure 6. Misalignment for CSS far in signal and single reflection sparse impulse response (_) NLMS algorithm, (_ _) memory efficient NLMS algorithm with compression by a factor of 2, (_.) memory efficient NLMS with compression by a factor of 4.

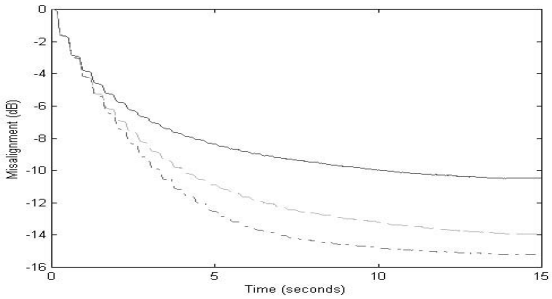

Figure 7. Misalignment for CSS far in signal and multiple reflection dispersive impulse response: (_) NLMS algorithm, (_ _) memory efficient NLMS algorithm with compression by a factor of 2, (_.) memory efficient NLMS with compression by a factor of 4.

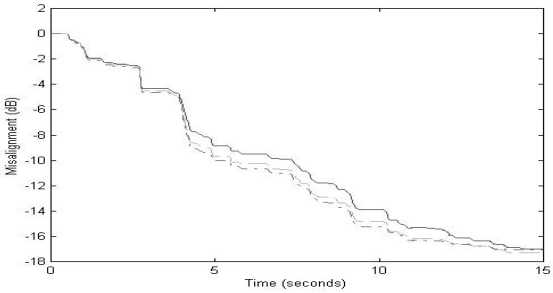

Figure 8. Misalignment for Speech far in signal, other parameters are same as Figure 6.

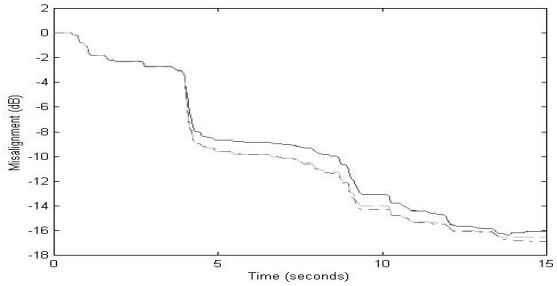

Figure 9. Misalignment for Speech far in signal, other parameters are same as Figure 7.

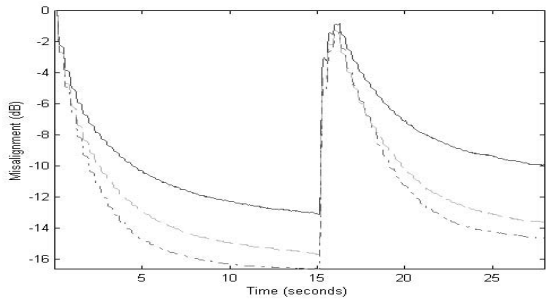

Figure 10 Misalignment during echo path change. The echo path changes from impulse response Fig 4(a) to Fig 4(b) at 15 seconds. The input is CSS signal.

4. CONCLUSION

We presented a memory efficient echo cancellation algorithm. This algorithm can reduce the memory requirements for storing echo canceller coefficients across frames in a VoIP media gateway by a factor of 2 to 4. Further the algorithm when combined with standard NLMS also improves the convergence rate of the echo canceller over the standard NLMS algorithm.

5. REFERENCES

[1] http://www.intel.com/network/csp/solutions/mediagateway/

[2] Bellamy, J., *Digital Telephony*, John Wiley & Sons, 2000.

[3] International Telecommunication Union, ITU-T G.168 Standard, 2000.

[4] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall, 1996.

[5] D. L. Duttweiler, "Proportionate normalized least mean square adaptation in echo cancellers," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 508-518, Sept. 2000.

[6] S. L. Gay, "An efficient, fast converging adaptive filter for network echo cancellation," *Proc. Assilomar Conf*, Nov. 1998.

[7] J. Benesty and S. L. Gay, "An Improved PNLMS Algorithm," *Proc. Intl. Conf on Accoustics, Speech, and Signal Processing*, pp. II-1881-II-1884, 2002.