Audio Engineering Society
Convention Paper

Presented at the 140th Convention
2016 June 4–7 Paris, France

Low complexity, software based, high rate DSD modulator using Vector Quantification.

Thierry Heeb[1], Tiziano Leidi[2], Diego Frei[3], and Alexandre Lavanchy[4]

[1] Research Scientist, ISIN - SUPSI, Galleria 2, CH-6928 Manno
thierry.heeb@supsi.ch

[2] Research Scientist, ISIN - SUPSI, Galleria 2, CH-6928 Manno
tiziano.leidi@supsi.ch

[3] Research Scientist, ISIN - SUPSI, Galleria 2, CH-6928 Manno
diego.frei@supsi.ch

[4] CEO, Engineered SA, Avenue des Sports 28, CH-1400 Yverdon-les-Bains
alavanchy@engineered.ch

ABSTRACT

High rate Direct Stream Digital (DSD) is emerging as a format of choice for distribution of high-definition audio content. However, real-time encoding of such streams requires considerable computing resources due to their high sampling rate, constraining implementations to hardware based platforms. In this paper we disclose a new modulator topology allowing for reduction in computational load and making real-time high rate DSD encoding suitable for software based implementation on off-the-shelf Digital Signal Processors (DSPs). We first present the architecture of the proposed modulator and then show results from a practical real-time implementation.

1. INTRODUCTION

Direct Stream Digital (DSD) is a high-resolution audio data format initially proposed by Sony and Philips in the late 90s for the Super Audio CD (SACD) optical disc. It is based on Pulse Density Modulation (PDM) and consists in a stream of single bit data sampled at high rate. For SACD, a sampling rate of 2.8224 MHz (64 x 44.1 kHz) was selected, providing high Signal to Noise Ratio (SNR) and wide bandwidth at the cost of a steeply raising noise floor above 20kHz. This raise results from the aggressive noise-shaping provided by the high-order delta-sigma modulator used for PDM generation. Despite its promises for high-resolution audio, the limited commercial success of SACD did constrain the DSD format to the audiophile community, with only marginal awareness of the format's existence among the general public.

Recent advances in dematerialized music distribution have brought DSD back to the front scene. Several music download sites are now offering a growing

catalog of DSD encoded high-resolution audio tracks, fueling consumer interest for the DSD format. As downloads are not constrained by the capacity of optical discs, higher DSD sampling rates can be used, lowering constraints on the modulators and pushing the steep noise floor raise away from the audio band. Sampling rates of 5.6448 MHz (128 x 44.1 kHz) or 11.2896 MHz (256 x 44.1 kHz) are now commonly supported by playback hardware such as USB Digital to Analog Converters.

One drawback of the DSD audio format is that any digital signal processing done on a DSD stream (except for a pure delay) destroys its one bit nature. Hence re-encoding is required, even after simplistic processing such as gain control. This causes a serious challenge in terms of computing resources for real-time systems, especially for high rate DSD at 128 or 256 x 44.1 kHz, calling for hardware based (ASIC or FPGA) implementation of the modulators and resulting in limited flexibility and added cost. This paper discloses a novel modulator topology addressing these issues and suitable for real-time implementation on off-the-shelf programmable Digital Signal Processors (DSPs). In the following sections, we present the key ideas leading to the proposed modulator architecture followed by the results of a practical implementation on a standard DSP. An introduction to the simulation infrastructure developed for the design of the new modulator architecture is also provided.

2. DELTA-SIGMA MODULATION

2.1. Background

Delta-sigma modulators [1] are widely used for PDM generation. The general structure for such a modulator is shown in figure 1. The output of the modulator is subtracted from the input signal and passed through the loop filter H before being sent to the quantizer Q.

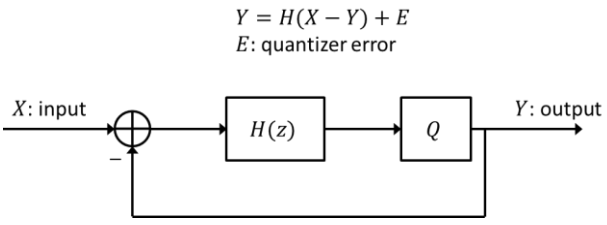

Figure 1: Delta-sigma modulator

Provided that the filter $H(z)$ exhibits a delay of at least one sample (for causality reasons) and that the quantization error E can be approximated by white noise, the system can be described by following equations:

$$Y(z) = STF(z)X(z) + NTF(z)E(z) \qquad (1)$$

$$STF(z) = \frac{H(z)}{1+H(z)} \qquad (2)$$

$$NTF(z) = \frac{1}{1+H(z)} \qquad (3)$$

where:

- $STF(z)$ is the Signal Transfer Function (STF) of the system, and
- $NTF(z)$ is the Noise Transfer Function (NTF) of the system

Following diagram shows a typical 5th order loop filter, implemented as a feed-forward topology with dual resonators. This topology is used for the modulators presented throughout this paper. Note that all integrators are of the delaying type, ensuring that $H(z)$ exhibits at least one sample delay. The resonators have a single sample delay, allowing for placement of the corresponding poles on the unit circle to guarantee stability of the Noise Transfer Function $NTF(z)$.

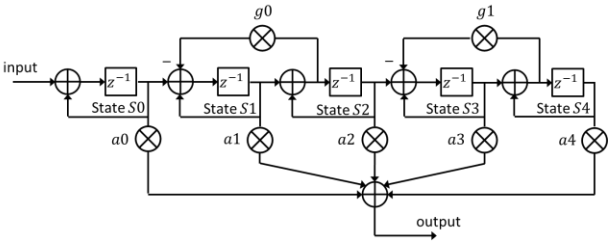

Figure 2: $H(z)$ loop filter implementation

2.2. State-space representation

Considering the vectors $A=(a_0,a_1,a_2,a_3,a_4)^T$, $S(n)=(S0(n),S1(n),S2(n),S3(n),S4(n))^T$, $FB(n)=(Y(n),0,0,0,0)^T$ and $IN(n)=(X(n),0,0,0,0)^T$, the modulator's operation can be described in the state-space domain by following matrix/vector operations:

$$S(n+1)=M\cdot S(n)+IN(n)-FB(n) \qquad (4)$$

$$Y(n+1)=Q\left[A\cdot S(n+1)\right] \qquad (5)$$

where

$$M=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1-g_0 & -g_0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1-g_1 & -g1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

is the state transition matrix and $Q[\ ]$ represents the quantization function.

2.3. Advanced modulation algorithms

Over the last decade, alternative topologies such as Trellis (see [2], [3] and [4]) or Look-Ahead ([5], [6] and [7]) modulators have been presented, offering greatly enhanced performance in terms of Signal to Noise Ratio (SNR) and/or NTF corner frequency. Instead of performing instantaneous, per sample quantization, these algorithms select their output among a number of output candidates, based on the optimization of a Cost Function (CF). Following diagram shows the operating principle of such a modulator.

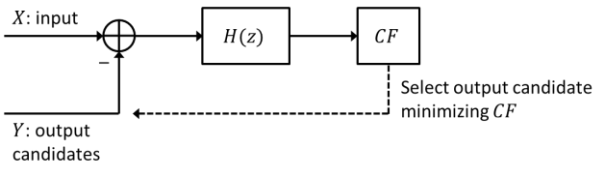

Figure 3: Modulator topology based on Cost Function

In a CF based modulator, output candidates are subtracted from the input signal and passed through a weighting filter $H(z)$. The output of this filter is then processed by the CF to select the output candidate minimizing its value. Typical output candidates are streams of tens to thousands of bits which inevitably introduce a corresponding delay into the system. Considering a CF based modulator maintaining K output candidates, each new input sample requires K computations of the loop filter and the CF, plus sorting among candidates. Hence its computational load is significantly higher than the one of a standard modulator. Typical values for K are between 4 and 32.

2.4. Requirements for high resolution audio and computing load estimations

In Reference [8], Stuart and Craven explore the peak level to back-ground noise ratio of high-definition recorded audio signals. These two signals tend to superpose above 45 to 50 kHz at a level around -100 dB to -110 dB. In order to faithfully reproduce the audio signal, noise introduced by the modulation process should not exceed this value. Due to their steep raise in noise floor above 20 kHz, standard delta-sigma modulators operating at 2.8224 MHz (DSD64) fail to meet these requirements. On the other hand, standard modulators running at 5.6448 MHz (DSD128) or 11.2896 MHz (DSD256) can easily be designed to come close to or exceed them. With a sufficiently large set of output candidates, advanced modulators, such as Trellis or Look-Ahead modulators, are able to meet or come close to these requirements at 2.8224MHz already, showing their clear advantage in terms of performance at a given sampling rate. Following table shows a summary of performance and computational load estimations for both standard and advanced modulators at different rates. The computational load for a standard modulator running at 2.8224 MHz is normalized to 1. Advanced modulators are supposed to be implementable at 4 times this load [9].

Modulator	Standard delta-sigma modulator		
Rate	DSD64	DSD128	DSD256
Performance	-	+	++
Load	1	2	4

Modulator	Advanced modulator		
Rate	DSD64	DSD128	DSD256
Performance	+	++	+++
Load	4	8	16

Table 1: Performance and load

Given that standard delta-sigma modulators match (or come close to) the requirements of high-definition recordings for DSD128 and DSD256, it is interesting to explore if gains in computational load can be obtained for these high rate DSD formats. Advanced modulators clearly show an advantage in terms of performance for a given DSD rate, but their computational load makes them less attractive for real-time implementation.

3. VECTOR QUANTIFICATION (VQ) MODULATOR

3.1. Modified modulator

Inspired by CF based topologies and [10], we propose a modified modulator as shown in the diagram below. Instead of applying the weighting filter $H(z)$ to the difference between the input signal X and the output candidates Y, both paths have their own weighting filter and the cost function is applied to the difference of the filters' outputs. Upon selection of the best output candidate, the states (integrators) of the weighting filter of the input path are updated accordingly.

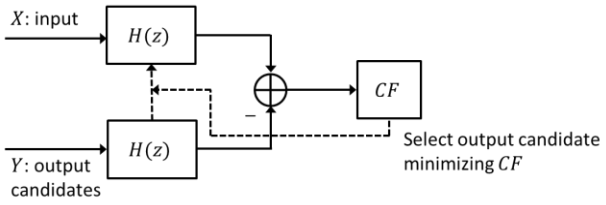

Figure 4: Modified, separated path modulator based on Cost Function

The primary benefit of this approach is to isolate the computation of the weighted contribution of the output candidates from the input signal (and past outputs) contributions. In other words, contributions from output candidates can be computed off-line and this is a key element of the new modulator topology. In addition, weighted paths for both input and output candidates are shaped by the low-pass characteristic of $H(z)$. Note that in the case of the CF being implemented as the magnitude of the current sample and using the filter $H(z)$ shown in figure 2, this topology can be made equivalent to the standard delta-sigma modulator of figure 1.

The idea behind Vector Quantization is to be able to produce multiple output bits at each processing round of the modulator. More precisely, at each processing round, an output vector $\overline{Y} = (y_1, y_2, ..., y_N)$ is produced, corresponding to N input samples (N is called the Block Length (BL) of the modulator). By doing so, the operation rate of the modulator is effectively reduced by a factor N. Provided that the complexity of the reduced rate modulator is less than N times the complexity of the full rate modulator, VQ allows for a reduction in computational load of the system. Given that the target platforms for the new modulator are programmable DSPs with single cycle full precision Multiply-Accumulate (MAC) instructions instead of hardware platforms such as ASICs or FPGAs, quantization of filter coefficients to the sum of a few powers of 2 is not required for optimal implementation. If output candidates are limited to the possible outputs of a data block, efficient implementation of the topology is possible.

Following diagram shows the operating principle of a VQ modulator. The input signal sampling rate is Fs.

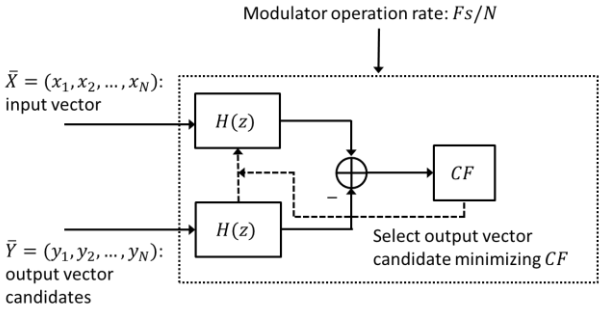

Figure 5: VQ modulator topology

3.2. Trajectories

Let's consider a VQ modulator with a Block Length of N. For a given state $S = (S0, S1, ..., SL)$ of the filter H and input data block $\overline{X} = (x_1, x_2, ..., x_N)$, we define the Path $P(\overline{X}, S)$ as the vector $(p_1, p_2, ..., p_N)$ of corresponding weighting filter output samples. A Trajectory $T(\overline{X}, S)$ is defined as a continuous time approximation of a Path. More

precisely $T(\overline{X},S)$ is a real valued function over the interval $[1;N]$ such that $\left|T(\overline{X},S)(n)-p_n\right|<\varepsilon$ for $n=1,...,N$, where ε is a (small) positive number. An example of a Trajectory with $\varepsilon=0$ is given by the N^{th} order polynomial approximation of the points (n,p_n). For sufficiently large ε, the linear regression of the points (n,p_n) can also be considered as a Trajectory of the system. The goal is to find low complexity Trajectories with sufficiently small ε to get cheap, reliable approximations of the system's behavior over its Block Length

Given that the filter H is a linear time invariant system, Paths are linear with respect to filter states $S0,S1,...,SL$ and input data block $\overline{X}$. Let S_i define the weighting filter state where all states are 0, except $Si=1$ for $i=0,1,...,L$. Assuming existence of low complexity approximations for $T(\overline{0},S_i),i=0,1...,L$ and $T(\overline{X},0)$, then

$$T(\overline{X},S)=\sum_{i=0}^{L} Si\ T(\overline{0},S_i)+T(\overline{X},0) \qquad (6)$$

is a low complexity approximation of $P(\overline{X},S)$. If $T(\overline{0},S_i),i=0,1...,L$ and $T(\overline{X},0)$ can each be expressed as polynomials of order R, Equation (6) can be rewritten as

$$T(\overline{X},S)=\sum_{i=0}^{L} Si\ \sum_{j=0}^{R} q_{ij}t^j+\sum_{j=0}^{R} v_j t^j=$$

$$\sum_{j=0}^{R}(v_j+\sum_{i=0}^{L} Si\ q_{ij})\ t^j \qquad (7)$$

As an example, consider a 5th order VQ modulator implementing the filter $H(z)$ shown in figure 2, and using a block length of $N=4$. Following picture shows the Paths $P(\overline{0},S_i),i=0,...,4$.

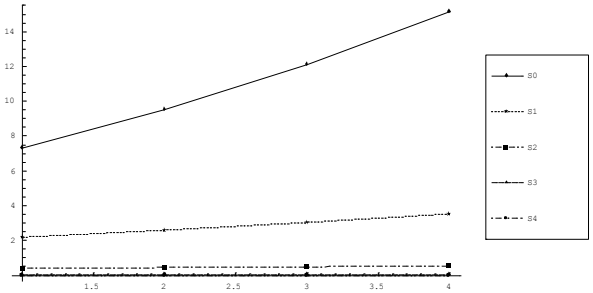

Figure 6: State paths (S0, S1, S2, S3 and S4)

Using a best fit algorithm, 2nd order polynomial Trajectories can be built for Paths $P(\overline{0},S_i),i=0,...,4$. Following table shows the Paths values and the related Trajectories relative error values.

Time	t = 1		t = 2	
State	Value	Error [%]	Value	Error [%]
S0	7.334	-0.029	9.524	0.068
S1	2.190	-0.004	2.591	0.011
S2	0.402	0.001	0.445	-0.002
S3	0.0434	0.0003	0.0453	-0.0008
S4	0.0020	-0.0002	0.0019	0.0006

Time	t = 3		t = 4	
State	Value	Error [%]	Value	Error [%]
S0	12.116	-0.053	15.152	0.014
S1	3.036	-0.009	3.526	0.003
S2	0.490	0.002	0.537	-0.001
S3	0.0472	0.0008	0.0490	-0.0003
S4	0.0019	-0.0007	0.0018	0.0002

Table 2: State Paths values and 2nd order polynomial Trajectories relative errors

Similar continuous-time extensions (Trajectories) for the output candidates $\overline{Y}=(y_1,y_2,...,y_N)$ passed through the filter H can be computed. Given the system's linearity in terms of filter states S, it is sufficient to consider Trajectories with initial state $S=0$. As the number of output candidates is limited (2^N candidates at most), results may be stored in a read-only table (or computed once at start-up) if N is not too large. Hence, getting Trajectories for output candidates reduces to simple table look-ups. Following picture shows the different output candidate Paths for a VQ modulator with Block Length $N=4$.

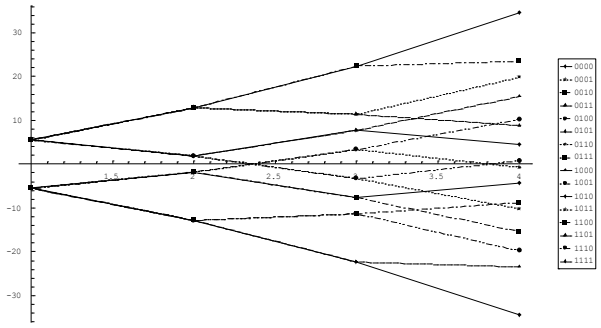

Figure 7: Output Candidates Paths ($N = 4$)

3.3. Quantization

Quantization is determined by the minimization of the Cost Function (CF) on the set of available output candidates. Given the concept of Trajectories which provide continuous-time extensions of the system's and output candidates' Paths, various types of Cost Functions can be considered, all representing some form of pattern matching between the system and the output candidates Trajectories.

For instance, if both system and output candidates Trajectories are given by (possibly piece-wise) polynomials, the Cost Function can be implemented as a closed-form analytical expression such as the L^2 norm of the Trajectories' difference (where T_n is the Trajectory of the n^{th} output candidate):

$$CF(T(\overline{X},S),T_n) = \sqrt{\int_1^N (T(\overline{X},S)(t) - T_n(t))^2 dt} \quad (8)$$

Alternatively, Cost Functions, which only compare Trajectories on a finite set of discrete points, may be used. Note that given the continuous-time nature of Trajectories, these discrete points don't need to be aligned with the sampling points of the input signal. An example of such a Cost Function is the sum of squared differences' magnitudes over $K+1$ discrete points:

$$CF(T(\overline{X},S),T_n) = \sum_{k=0}^{K} (T(\overline{X},S)(t_k) - T_n(t_k))^2 \quad (9)$$

In order to keep computational load low, it is tempting to consider very simple Cost Functions. One such example is the *mean* function defined by

$$mean(T(\overline{X},S),T_n) = \left| \sum_{k=1}^{N} T(\overline{X},S)(t) - \sum_{t=1}^{N} T_n(t) \right|.$$

Whilst this CF offers good performance for small modulation indexes (i.e. for small input signals), it results in modulator instability for high level input signals. The reason for this is that the *mean* function destroys any phase information present in the Trajectories. Hence CF should be designed to preserve (at least part of the) phase information.

If $T(\overline{X},S)$ can be discretized to a finite number of different values, computation of the Cost Function can be approximated by a table look-up, making it very efficient in terms of actual implementation. For instance, consider the case where $N = 4$ and the system's Trajectories are defined by 2nd order polynomials. Coefficients of the 2nd order polynomials $T(\overline{X},S) = b_0 + b_1 t + b_2 t^2$ can be computed using Equation (7) and quantized to B_i bits respectively ($i = 0,1,2$). The Cost Function can then be approximated by accessing a $2^{(B_0+B_1+B_2)}$ entries look-up table, containing the indexes of the optimal output candidates. This table is called the Quantization Table (QT). Granularity of the QT must be fine enough to properly resolve the different output candidates.

Dithering must generally be applied to avoid correlation between quantization noise and input signal content. It can be applied directly at the point of the discretization of the system's Trajectories parameters (discretization of coefficients b_i to B_i bits in the example above) or values. Alternatively, it may be integrated into the Quantization Table. Instead of storing only the optimal output candidate for a given discretized Trajectory $T(\overline{X},S)$, the QT may be extended to store the indexes of the r $(r>1)$ best output candidate. Dithering can then be implemented by using a random number generator with range $0,...,r-1$ to select one among the r best output candidates. Mechanisms to constrain the selection depending on input signal level and/or internal modulator states may be required to guarantee modulator stability.

3.4. Efficient state-space implementation

State-space representation provides the foundation for efficient implementation of the proposed VQ modulator.

Starting from the state-space equations of a standard modulator, we observe that under the condition of constant '0' feedback, the system's state at time $n+N$ can be computed by iterating Equation (4) N times. If the input data vector $\overline{X}=(x_1,x_2,...,x_N)$ can be considered as constant (i.e. $x_n=x_1$ for $n=1,...,N$), this can be expressed as:

$$S(n+N)=SEM\cdot S(n)+IEV(n) \qquad (10)$$

where $SEM=M^N$ is designated as the State Evolution Matrix (SEM) and $IEV(n)=\sum_{i=1}^{N}M^{i-1}\cdot IN(n)=\sum_{i=1}^{N}M^{i-1}\cdot(x_1,0,...,0)^T$ is called the Input Evolution Vector (IEV).

For an output candidate with index n, let $OSE_n=(ose_0,ose_1,...,ose_L)^T$ be the vector of state values obtained when running this output candidate through the filter H. Based on these definitions, the operation of the VQ modulator can be described by the following algorithm:

1. Compute $T(\overline{X},S)$ and discretize its parameters or values.
2. Use discretized values to find optimal output candidate index n_0 (with dithering) by QT look-up.
3. Update state values using Equation (10).
4. Correct state values for the optimal output candidate using $S(n+N)=S(n+N)-OSE_{n_0}$.
5. Limit state values (to control modulator stability).

In order to estimate the complexity of the proposed VQ modulation algorithm, it is interesting to compare it to the standard delta-sigma modulator algorithm described by equations (4) and (5). Let's consider a standard 5th order modulator implementing the filter of figure 2. For typical values of $K=2$ or 3 and $N=4$, complexity estimations result in about 40% savings compared to the standard modulator. This would render high rate DSD modulation suitable for software implementation on modern DSPs.

4. SIMULATIONS AND PRACTICAL IMPLEMENTATION

4.1. Simulations

With the goal of easing and accelerating the identification of design inefficiencies and development errors in the modulator, an infrastructure for automatic execution of large number of simulation runs has been set up. Need for efficient an simulation infrastructure was mainly motivated by the non-linear nature of the modulator system. Small variations in the definition of the Cost Function modify the modulator behavior significantly, with consequent risk of instabilities and poor performance. Extensive simulations allowed for converging to low-complexity, low-distortion modulators.

For this purpose, a development tool called Digital Stream Processing Environment (DSPE) [11, 12] has been adopted and customized with specific extensions specific for 1 bit modulator analysis. DSPE is a development tool provided by SUPSI for designing and implementing parallel stream-processing applications for multi-core processors and accelerators such as GPUs [www.systemdesigner.ch]. It is integrated with the Eclipse development platform. DSPE features a model-based domain-specific language and C/C++ source code generators that prove particularly useful at application prototyping and performance analysis. DSPE is released under the open-source Eclipse Public License (EPL).

In recent academic research projects [13], DSPE has been extended with functionality for performing computationally intensive simulations by means of dedicated support for scheduling batch executions of multiple application runs. This infrastructure has been exploited for performing the required modulator simulations. For this purpose, DSPE has been enhanced with specific software components for several variants of the modulator (in particular in terms of Cost Function) and for analysis of the resulting modulated data streams.

In particular, simulations have been used to optimize discrete Cost Functions according to Equation (9). Simulations revealed that selection of the discrete time points for Cost Function computation have a significant impact on modulator performance. Among others, it was found that Cost Functions operating properly at high input signal level may perform poorly when fed with small input signals or the other way round.

Following figure shows such a case where distortion appears for a -34dB input sine wave. Thanks to extensive simulation runs, simple (2 or 3 points), low distortion, discrete Cost Functions could be constructed and subsequently implemented for real-time applications.

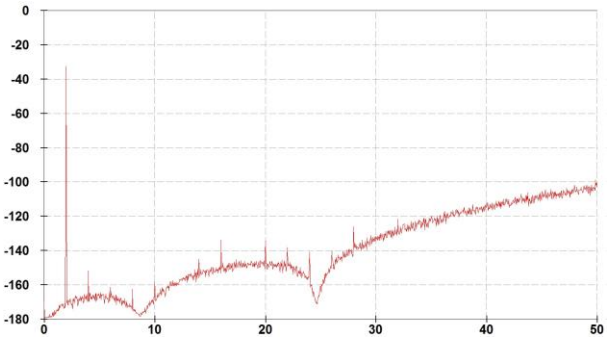

Figure 8: Typical small signal error for poorly defined Cost Function

4.2. Real-time implementation on ADSP-21489

In order to compare performances and computational loads, both the standard delta-sigma and the VQ modulator (with $N = 4$) algorithms have been ported to the Analog Devices ADSP-21489 DSP. The implementations have been optimized in Assembler and make extensive use of the processor's SIMD architecture to process stereo data streams. Real-time audio performances have been analyzed using the setup shown in the figure below:

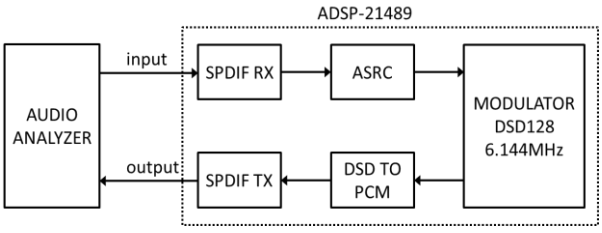

Figure 9 Test setup

The audio analyzer generates digital test signals sent to the DSP over SPDIF. The DSP recovers the PCM digital audio data and applies the modulation algorithms to produce stereo DSD streams. These streams are then processed by a high quality DSD to PCM converter before being sent back to the analyzer over SPDIF. Note that the test platform's architecture requires an Asynchronous Sample Rate Converter (ASRC) to be used on the DSP's SPDIF input and that modulators are effectively operated at multiples of 48 kHz. Following table shows some performance figures for DSD128 encoding.

Measurement	Standard delta-sigma	VQ modulator
SNR (20Hz-20kHz), unweighted	-126 dB	-126 dB
SNR (20Hz-20kHz), A-weighted	-129 dB	-129 dB
THD+N (20Hz-20kHz), 1kHz 0dB	-126 dB	-126 dB

Table 3: Audio performance figures

The graphs below show the low frequency spectrum of DSD128 modulation with a -3dB, respectively -80dB 1kHz sine wave input. Plots have been made using 8 times averaged 32'768 points FFTs. Results for DSD256 can be extrapolated by doubling the frequency.

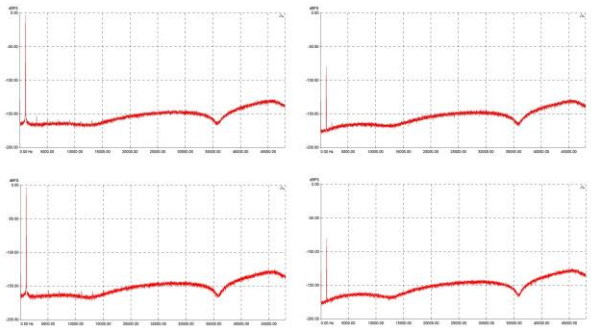

Figure 10 Modulator low frequency spectrum plots, standard modulator (top), VQ modulator (bottom)

The above results show that the performances of the VQ modulator are equivalent to those of the standard modulator. Computational load has been analyzed using both linear and statistical profiling. Linear profiling was used for dry algorithms analysis whereas statistical profiling was used to assess computational loads under real-world conditions. Results are shown below.

Setup	Standard delta-sigma	VQ modulator
DSD128, linear profiling	180.5 MIPS	108.5 MIPS
DSD256, linear profiling	361 MIPS	219 MIPS
DSD128, statistical profiling	183 MIPS	110 MIPS
DSD256, statistical profiling	365 MIPS	221 MIPS

Table 4: Computational load

The VQ modulator provides a close to 40% benefit in computational load with no noticeable impact on performance compared to the standard delta-sigma modulator, making it very attractive for software based implementation. In particular, VQ modulators allow for high-quality stereo DSD128 encoding at less than 110MIPS on modern DSP architectures.

5. CONCLUSIONS

This paper has presented a new, simple algorithm for high rate DSD encoding showing similar performance to standard delta-sigma modulation at significantly reduced computational load, making it suitable for software based implementation on off-the-shelf digital signal processors.

This advantage in terms of computational requirements can also be used to implement modulator enhancements while maintaining load similar to standard delta-sigma modulators. Taking SDPC (Sigma Delta Prediction Correction) [14] as an example, the load reduction provided by VQ modulators allows the two modulators required for SDPC (in its basic form) to consume only slightly more MIPS than a single standard modulator. Similarly, the VQ modulator also presents advantage for advanced topologies such as Trellis modulators. By replacing the standard modulators used for each output candidate, the computational load of advanced modulators can be reduced accordingly. In addition, VQ modulation allows for classification according to Cost Function of the 2^N possible evolutions of an output candidate to be realized by simple table look-ups in an extended Quantification Table, which could bring further reduction in computational load. In such an implementation, the individual VQ modulators can be considered as small, local Trellis over the 2^N possible evolutions of an output candidate.

Research on VQ modulators currently continues within our group with focus on achieving average pulse rate reduction and increasing the maximum stable modulation index to make the algorithm suitable for digital switching amplifier applications. We work on exploiting the reduced computational load of VQ modulators to apply both optimal and reduced pulse rate output candidates in parallel and to decide which path to follow depending on evolution of the internal modulator states over the following rounds. Although this research is still in its early stages, it already shows promising results.

6. ACKNOWLEDGEMENTS

This work has been sponsored by the on-going grant 16890.1PFIW-IW of the Swiss federal Commission for Technology and Innovation (CTI) and has been realized in collaboration between the University of Applied Sciences of Southern Switzerland (SUPSI) in Manno and Engineered SA in Yverdon-les-Bains, Switzerland.

7. REFERENCES

[1] S.R. Norsworthy, R. Schreier and G.C. Temes, “Delta-Sigma Data Converters, Theory, Design, and Simulation”, IEEE Press, Wiley-Interscience, 1997, ISBN 0-7803-1045-4.

[2] H. Kato, “Trellis Noise-Shaping Converters and 1-bit digital audio”, presented at 112th AES Convention, Munich, Germany, 2002 May 10-13.

[3] P. Harpe, D. Reefman and E. Janssen, “Efficient Trellis-type Sigma Delta Modulator”, presented at 114th AES Convention, Amsterdam, The Netherlands, 2003 March 22-25.

[4] E. Janssen and D. Reefman, “Advances in Trellis based SDM structures”, presented at 115th AES Convention, New-York, United States of America, 2003 October 10-13.

[5] J.A.S. Angus, “Tree Based Lookahead Sigma Delta Modulators”, presented at 114th AES Convention, Amsterdam, The Netherlands, 2003 March 22-25.

[6] J.A.S. Angus, “Efficient Algorithms for Look-Ahead Sigma-Delta Modulators”, presented at 115th AES Convention, New-York, United States of America, 2003 October 10-13.

[7] M.O. Hawksford, "Parallel Look-Ahead Digital SDM with Energy-Balance Binary Comparator", J. Audio Eng. Soc., vol.56, No 12, pp. 1069-89, 2008 December.

[8] J.R. Stuart and P.G. Craven, "A Hierarchical Approach to Archiving and Distribution", presented at 137th AES Convention, Los Angeles, United States of America, 2014 October 9-12.

[9] E. Janssen and D. Reefman, "DSD compression for recent ultra high quality 1-bit coders", presented at 118th AES Convention, Barcelona, Spain, 2005 May 10-13.

[10] US Patnet US 7'292'171, "Method and device for the version of digital signals with heterogeneous formats and application thereof to the digital amplification of audio signals"

[11] T. Leidi, T. Heeb, M. Colla and J.-P. Thiran, "Model-Driven Development of Audio Processing Applications for Multi-Core Processors", presented at 128th AES Convention, London, United Kingdom, 2010, May 22-25.

[12] T. Leidi, T. Heeb, M. Colla and J.-P. Thiran, "Event-Driven Real-Time Audio Processing with GPGPUs", presented at 130th AES Convention, London, United Kingdom, 2011, May 13-16.

[13] T. Leidi, G. Scocchi, A. Ortona, L. Grossi, S. Pusterla , C. D'Angelo and J.-P. Thiran, "Computing effective properties of random heterogeneous materials on heterogeneous parallel processors", in Computer Physics Communications, Elsevier, 2012. http://dx.doi.org/10.1016/j.cpc.2012.06.010

[14] D. Reefman and E. Janssen, "Enhanced Sigma Delta Structures for Super Audio CD Applications", presented at 112th AES Convention, 2002, Munich, Germany, 2002 May 10-13.