

Invention Disclosure

Scalable MPEG Layer II Audio Coder

Invented By

Victor Ramamoorthy

Invention Occurred during: February 1997 when Victor Ramamoorthy was employed by Adaptive Media Inc. Currently Victor Ramamoorthy is not employed by Adaptive Media Inc. and he can be reached at vrm@worldnet.att.net or by calling 510-417-8517.

Filing Date: 28 October 1997

Table of Contents

Invention Disclosure	1
Table of Contents	2
1.0 Requirements of Audio Sigma Stream	3
1.1 Invention Claims	3
2.0 Starting Point.....	3
3.0 MPEG System	4
4.0 Adaptive Media Audio Decoder.....	5
5.0 Architecture of Audio Sigma Stream	6
6.0 Structure of Vector Quantization.....	9
7.0 8-Dimensional Vector Quantizer 1 bit/sample Codebook (32 KHz sampling frequency)	10
8.0 Performance of VQ design.....	15
9.0 Program Listings	16
9.1 Vector Quantizer Encoder and Decoder Program	16
9.2 Prediction Difference Generation for Vector Quantization.....	20
9.3 Batch File to run Adaptive Media Transcoder	21
10.0 References	24

1.0 Requirements of Audio Sigma Stream

The requirements of Adaptive Media Audio Sigma Stream are the following:

- (1) Sigma streams should be additive so that higher quality streams can be built from component lower quality streams
- (2) The storage efficiency of the entire scheme should be very high; the additional storage required for scalable stream generation should be very small,
- (3) There should be no penalty in terms of audio quality degradation at any stage of stream composition
- (4) Computational burden imposed on the client should be exceedingly small,
and
- (5) The system should lend itself to new modifications and proprietary innovations.

In the scheme described in this document achieves all these requirements simultaneously. As far as our knowledge goes, there is no such system published or built elsewhere.

1.1 Invention Claims

1. Novel storage scheme for audio so that low bit rate streams can be read off from the stored high bit rate audio file.
2. Novel streaming scheme for audio so that low bit rate stream can be parsed from the transmitted high bit rate stream.
3. Very high compaction and efficiency in the storage and transmission format.
4. No audible quality degradation due to the proposed method.
5. No additional complexity in terms of computation or hardware logic. Only an extra addition is required at the client (receiver) side.
6. Makes use of MPEG layer II coding which is inherently not scalable.
7. Vector quantizer used in the system only represents a simple processing unit. Additional perceptually based algorithms can be incorporated as a distortion measure in the vector quantizer.

2.0 Starting Point

The starting point of Sigma streams is the MPEG-1 Layer 2 Audio coder. This encoder accepts audio inputs sampled at 32 KHz, 44.1 KHz, or 48 KHz and digitized to 16 bits/sample. MPEG audio coder is a subband coder with perceptual quantization strategy by incorporating a human auditory hearing system model. There are 32 subbands used in the system. Layer 1 and 2 use a linear quantizer to quantize the subband signals. The quantization is done in such a way that the quantization noise is inaudible to the human listener.

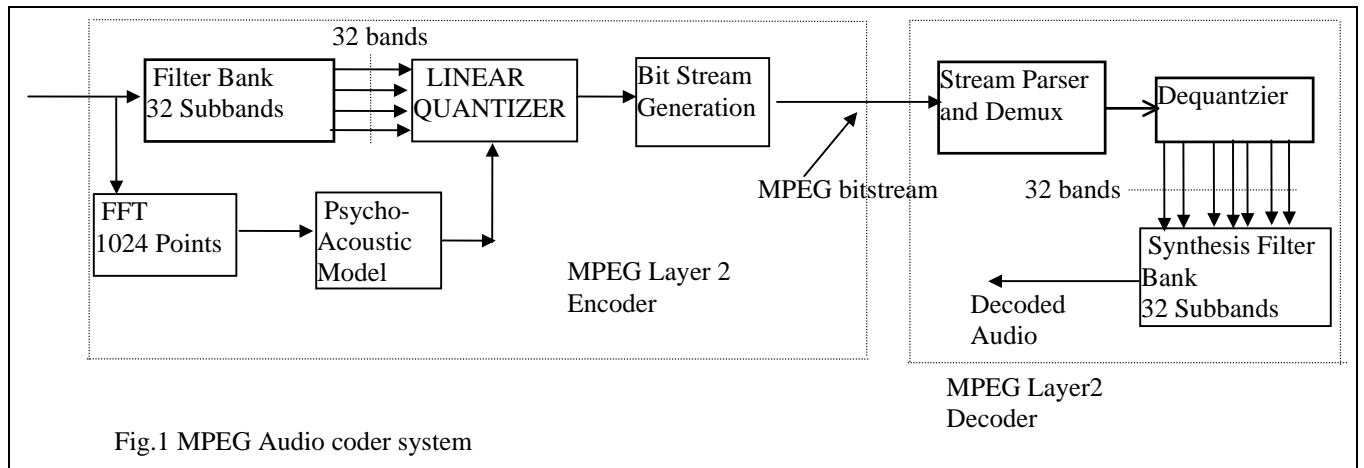
The hearing model assumes a simplified classification of an audio input as tonal and non-tonal components. This is achieved by performing a narrow band signal analysis with a 1024-point FFT (or 512-point FFT for layer 1) The output of the signal spectral analysis is used to determine the relevant tonal and non-tonal noise maskers in the actual input signal. It is well known from the psycho-acoustic research that human auditory system's inability to hear quantization noise under conditions of auditory masking which usually occurs when a strong audio signal is surrounded by weaker signals in the temporal and spectral domain. There could be two different conditions: tone masking noise and noise masking the tone. The amount of masking perceived is a function of signal's tonality, its temporal/frequency position and its loudness. Further human auditory system has a limited frequency-dependent resolution. Hence the incoming audio signal is divided into frequency bands close to the "auditory critical bands", different

masking thresholds are computed for each band to allocate different number of quantization bits. The bit allocation is dynamic and is input signal dependent. The details of perceptual quantization are not relevant in the present context. Interested reader is referred to the seminal work of Jim Johnston [1] and the tutorial by Davis Pan[2].

3.0 MPEG System

The reason for selecting Layer 2 is the presence of **linear quantization** of 32 subbands in the system. The other reasons include simplicity of implementation, absence of variable length codes and bit accumulation as in the bit reservoir model of Layer 3. The MPEG audio encoder and decoder system is shown in fig.1.

The encoder in the system is more complex than the decoder. The encoder does spectral analysis with the filter bank as well as a 1024-point FFT . Then by the use of the psycho-acoustic model, the subband signals are linearly quantized so as to fit the requirement of bit rate and audio quality. The quantized subband signals are then packed into a bit stream and sent over to the decoder. The decoder has a stream parser and demultiplexer to undo the multiplexing operation performed at the encoder. The received 32 band signals are then dequantized and fed into a synthesis filter bank.



The output of the filter bank forms the decoded audio output. By the very construction of MPEG layer 2 bit stream, it is desirable to tap into the dequantizer output before it is fed into the synthesis filter bank. A suitable construction for Adaptive Media Audio Decoder is shown in fig.2.

Adaptive Media Decoder is essentially derived from the generic MPEG decoder. The decoder construction is based on the idea of successive improvements (or simply *Bootstrapping*) starting from a base bit rate stream. The MPEG base bit stream is denoted by S_0 . As an example, the base stream could be a 32 Kilobits/sec MPEG stream. If S_0 is alone available at the decoder, then the decoded output will be exactly same as that of a conventional MPEG output. On the other hand, if another stream, say S_1 is also available, then the output of the Adaptive Media decoder would be at a higher quality corresponding to the sum of the streams $S_0 + S_1$. The stream S_1 is called an enhancement layer stream. There could be multiple enhancement streams as shown in fig.2. In other words, by summing all the enhancement layers with the base stream, the decoder delivers the highest quality possible with a combined bit rate of all the enhancement layers and the base stream. Intermediate levels of quality can be achieved by only including a few of the enhancement layers. By construction, the layers are hierarchical and ordered. What is desired by the arrangement can be described as follows:

Decoded Quality of $S_0 \implies$ Decoded quality of a MPEG decoder with a bit rate R_0 of S_0 .

Decoded Quality of $S_0 + S_1 \implies$ Decoded quality of a MPEG decoder with a combined bit rate of $R_0 + R_1$, where R_1 is the bit rate of S_1 .

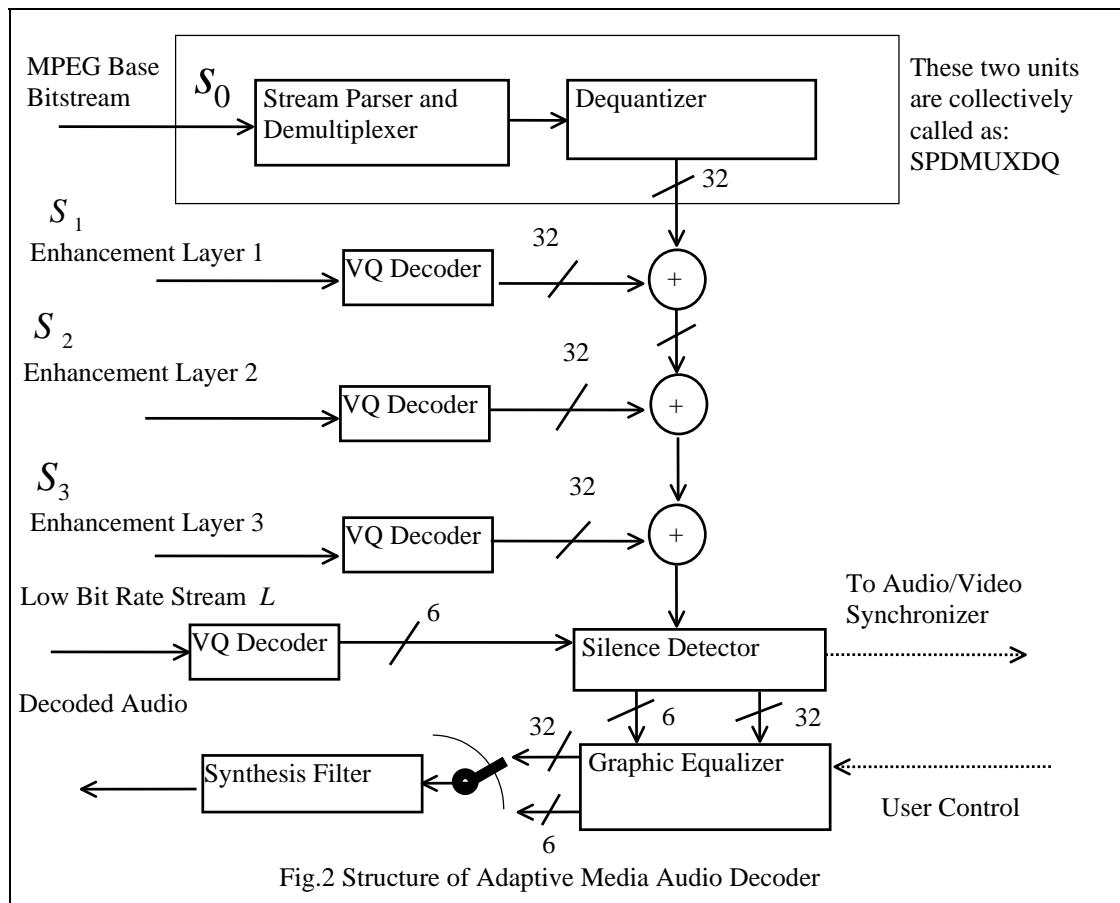
Decoded Quality of $S_0 + S_1 + S_2 \implies$ Decoded quality of a MPEG decoder with a combined bit rate of $R_0 + R_1 + R_2$, where R_2 is the bit rate of S_2 .

Decoded Quality of $S_0 + S_1 + S_2 + S_3 \implies$ Decoded quality of a MPEG decoder with a combined rate of $R_0 + R_1 + R_2 + R_3$, where R_3 is the bit rate of S_3 .

In the above the symbol “ \implies ” means that the audio was presented to the listener through a pair of generic multimedia PC speakers kept at a minimum distance of 24” from the listener in an uncontrolled office cubicle environment; further quality of the left hand side of “ \implies ” is deemed to be indistinguishable from that on the right side of “ \implies ”.

The above partitioning of quality and bit rate is exact and complete. That is the partitions add up exactly and there are no overheads. Such partitioning is known as *Sigma Partitioning*. Until now, there were no practical solutions that can realize or implement the idea of Sigma Partitioning. With Sigma Partitioning, it is possible to achieve scalability at no extra cost.

4.0 Adaptive Media Audio Decoder



The structure, shown in fig.2, consists of two other elements not directly related to scalability. The first one is the *Silence Detector* and the other is the *Graphic Equalizer*. The silence detector detects silences in the audio by detecting whether the input signals are within a bounded region or not. The graphic equalizer performs bass/treble tuning and volume control by multiplying the input with pre-assigned scale factors. The scale factors involved could be as simple as binary “on/off” switches. The design of the graphic equalizer is treated separately in section 10.0.

The streams S_0, S_1, S_2, S_3 belong to a medium bit rate family. In this family, S_0 is the base stream with the lowest bit rate and quality. Next level is achieved by including the stream S_1 along with S_0 . Since S_0 is a conventional MPEG stream, Stream Parser, Demultiplexer and Dequantizer are required to extract the quantized subband outputs from S_0 . The other streams S_1, S_2, S_3 are encoded by a vector quantizer and hence require Vector Quantizer Decoders. The output of each of VQ Decoders generate 32 subband difference outputs which are then added to the subband outputs generated from S_0 . The sum of all the outputs is then fed to the synthesis filter through the graphic equalizer and silence detector.

Yet another stream that is found in the decoder architecture of fig.2 is the Low Bit Rate Stream L . This stream contains low bit rate encoded sequence that is treated separately from the medium bit rate streams. As before, the low bit stream L also requires a VQ decoder; however, the output from this decoder consists only of six subband signals instead of 32. The low bit rate stream decoded outputs are also routed through the silence detector and graphic equalizer. The synthesis filter can take either the combined 32 subband signals of the medium rate hierarchical coders or the low bit rate coder signal consisting of only six subbands. If any of the enhancement layers are absent, the corresponding VQ decoders output zero values.

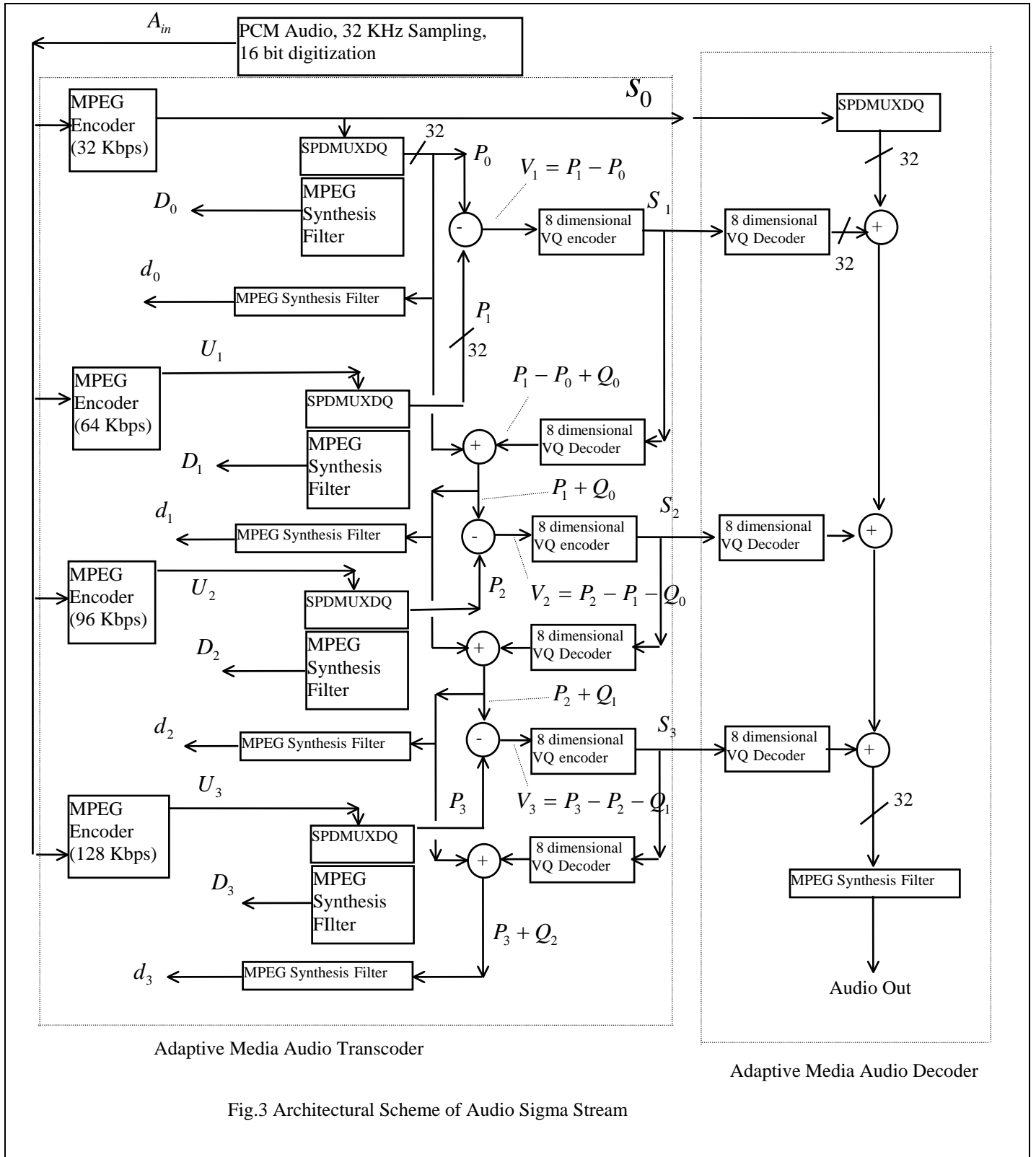
5.0 Architecture of Audio Sigma Stream

Fig.3 shows the architecture of Adaptive Media Audio Sigma Streams. The input A_{in} to the system is 16 bit linear PCM audio sampled at 32 KHz. The base level stream S_0 is generated by the MPEG encoder (layer 2) with a bit rate of 32 Kbps. In fig.3, the transcoder is shown on the left hand side and the Adaptive Media Decoder is shown on the right hand side.

The transcoder has a set of encoders and decoders connected as cross-feed loops. At the first level, the base stream S_0 is decoded with a conventional MPEG decoder. This conventional decoder is split as two parts: the first one is the stream parser, demultiplexer and the dequantizer called as SPDMUXDQ. The second part is the synthesis filter. The output of the conventional decoder with input S_0 is denoted as D_0 . The output from the first part SPDMUXDQ is called as P_0 . At the decoder side, the same P_0 is available as the output from SPDMUXDQ unit. If P_0 is fed to the synthesis filter, then the decoder will have an identical copy of D_0 denoted as d_0 . The transcoder then feeds the same audio input A_{in} to a 64 Kbps encoder which generates the compressed stream U_1 . Unlike S_0 , the compressed stream U_1 is not transmitted. However it is fed to the local decoder to generate the dequantized filter outputs P_1 and the decoded audio output D_1 . In order to generate the first enhancement layer, the difference $V_1 = P_1 - P_0$ is computed and quantized by an 8-dimensional VQ with a bit rate of 1 bit/sample. The VQ encoder output which is a stream of code book indices form the enhancement layer 1 stream called as S_1 . If the stream is fed to a corresponding VQ decoder, then the decoder output would contain $P_1 - P_0 + Q_0$, where the term Q_0 is the additive quantization noise arising in the vector quantization process. By the very construction of the decoder, the VQ decoder output is added with the base layer signal P_0 ; this means that base layer

stream S_0 and S_1 give rise to the audio output of d_1 at the decoder. This audio output d_1 is slightly different from the conventional MPEG decoder output of D_1 , the difference being due to the quantization noise Q_0 sent over to the synthesis filter in addition to P_1 .

Since the transcoder duplicates the situation at the decoder, exact cancelling of quantization noise is possible in each enhancement stage. As a result, the quantization noise for each stage does not add up. Each stage has only a single contamination with quantization noise. The construction of cross-feed loops guarantee elimination of quantization noise accumulation. The audio outputs D_1 and d_1 differ because of the quantization noise Q_0 . Similarly D_2 and d_2 differ because of Q_1 . D_3 differs from d_3 because of the quantization noise Q_2 .



6.0 Structure of Vector Quantization

Vector quantizer (VQ) has a performance advantage over scalar quantizers. Vector quantizers take into account of the hidden correlation structures in the input and thereby offer better performance..

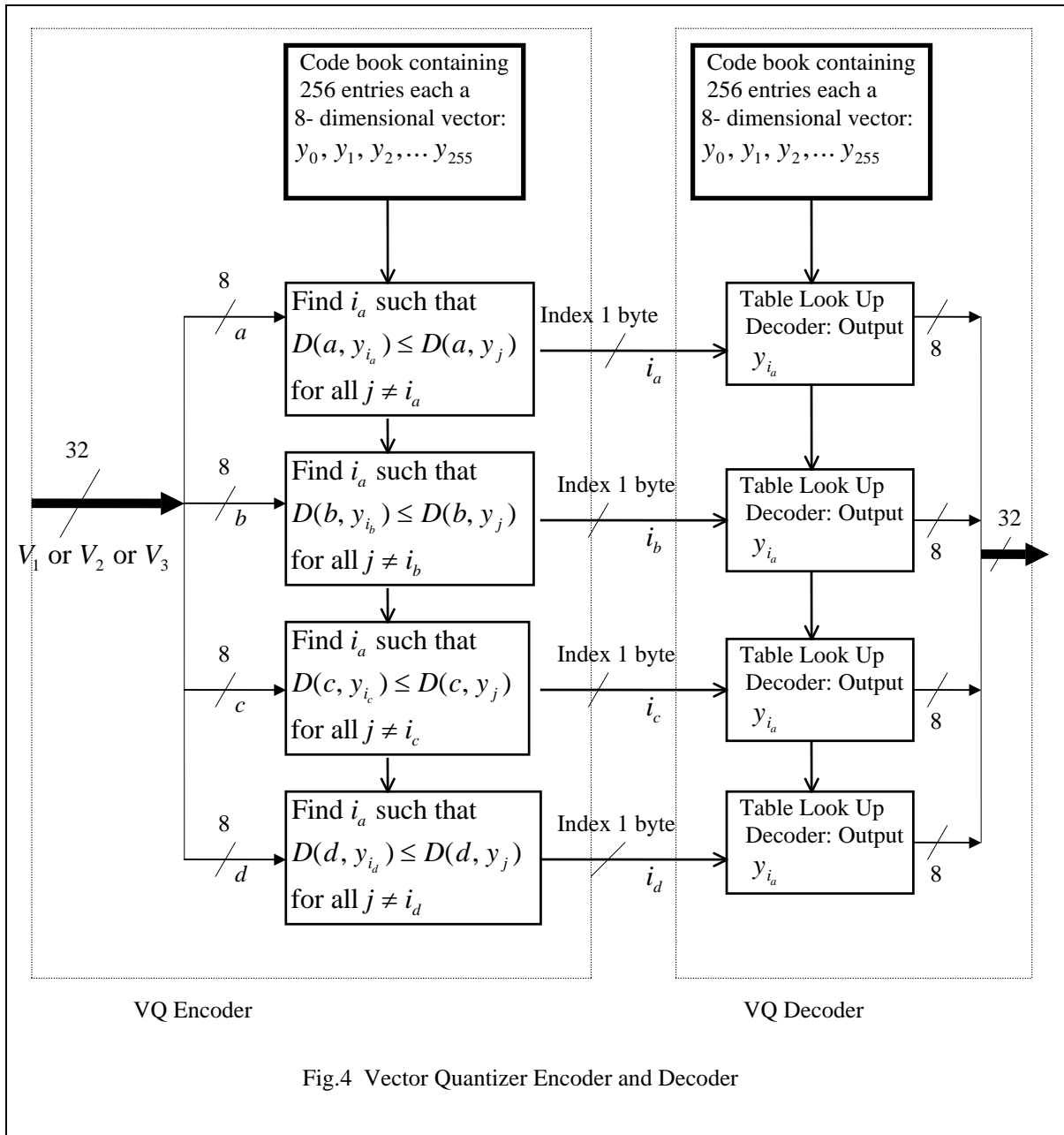


Fig.4 Vector Quantizer Encoder and Decoder

Fig. 4 shows an 8- dimensional 1 bit/sample VQ coder used in the high bit rate part of the Adaptive Media coder. To achieve 1 bit/sample rate, the VQ stores a codebook of 256 entries shown in section 7.0. Adaptive Media coders also employ another 6-dimensional 2 bit/sample VQ coder as detailed in section 8.0. In this section, attention is focused only on the 8-dimensional 1bit/sample VQ coder.

The input to the VQ is the difference signals V_1, V_2 , or V_3 , each a 32-dimensional double precision floating point vector. The 32-dimensional input is decomposed as four 8-dimensional vectors and are separately encoded. VQ encoding consists of finding the best match between the input 8-dimensional vector and the precomputed VQ codebook entries. There is only one codebook is maintained for each of the four VQ encoders that process the input 8-dimensional vectors. The criterion for best match is given by the distortion measure $D(.,.)$ between the input and the codebook entries. There are many possible ways of defining the distortion measure. Exhaustive simulations indicate that L2 norm is as good as other complex measures. L2 norm also is simpler to compute than measures like Itakura-Saito. In view of performance and simplicity, L2 norm is chosen. The codebook was generated by using a music sequence containing more than 5 million samples. A program to create the codebook using the splitting algorithm given in the Linde, Buzo and Gray paper referenced in [4]. After finding the best match, the index of the codebook entry is stored as the compressed value. Since the input is 8 floating point numbers and the output index is 1 byte, the bit rate is 1 bit/sample which translates to 32 Kbps because of 32000 Hz sampling used.

The decoder must store the codebook for table look up. As soon as it receives the compressed value, namely the index of the codebook entry, it outputs the corresponding codebook entry. For 1 bit/sample coding, the input to the VQ encoder is premultiplied by 32767.0 and rounded of as an 16-bit integer(or a "short" value with 2 bytes). This makes the codebook storage as $256 \times 8 \times 2 = 4096$ bytes.

The decoder complexity is minimal. All the decoder has to do is to perform a table look up. This makes it possible to have a low complexity decoder for the Sigma Stream decoding.

7.0 8-Dimensional Vector Quantizer 1 bit/sample Codebook (32 KHz sampling frequency)

The following table gives the codebook obtained by actual training with a long music segment containing suitable combinations of different speech and music.

<u>Index</u>	<u>Vector</u>							
0	0	0	0	0	0	0	0	0
1	-72	6	3	0	-1	0	0	0
2	-6	-3	-128	-8	-2	0	2	0
3	4	-33	13	-11	-12	22	2	-383
4	194	-161	51	8	2	0	3	-1
5	-36	-7	9	11	8	8	-10	371
6	-4	-10	148	-4	-2	2	3	0
7	-49	-12	1	-283	-22	-8	0	1
8	503	-27	16	4	10	-2	-1	6
9	-254	-188	-66	-35	-129	138	552	-318
10	327	-11	297	-73	540	46	-100	52
11	58	80	-124	100	132	-508	40	577
12	-3	9	-2	-9	231	29	0	-9
13	-5	-5	8	-1	-19	-271	-30	1
14	-148	185	15	0	3	-4	8	-4
15	4	-10	3	-21	-43	279	-13	7
16	-17	-115	-2	-2	-2	0	0	0
17	-226	141	210	416	72	20	-30	-23
18	-190	-174	-358	-131	42	2	-19	9
19	141	303	-65	120	5	-321	-442	-511
20	-75	-566	-153	41	661	442	206	203
21	-9	-54	-1	-16	70	-217	352	0
22	-565	-247	168	307	-210	403	-226	-115
23	-106	5	22	7	-350	5	-3	5

24	2	5	-50	216	-21	0	-1	0
25	-210	247	292	-596	266	175	197	-22
26	698	348	248	10	-70	-14	-24	18
27	-33	-9	33	-1	27	38	-411	-26
28	139	123	-194	-154	413	628	-679	315
29	37	265	-99	-164	434	-484	-228	24
30	11	439	387	-18	16	-12	0	2
31	69	39	52	111	-43	157	441	126
32	218	40	-16	-12	0	-1	-2	1
33	-267	-214	66	14	-4	2	4	3
34	303	-224	-121	-346	198	134	155	-43
35	429	7	0	-71	-68	311	-197	-615
36	-31	-534	226	354	-128	-58	69	-12
37	387	-105	-55	-96	-91	100	134	608
38	-108	121	362	-60	-1044	174	-38	-248
39	-457	-309	-250	-763	107	13	-8	-30
40	336	82	-88	104	-451	116	217	-113
41	-877	-31	53	-16	-19	-5	20	0
42	1048	-237	222	76	29	-29	34	10
43	223	-212	178	306	92	15	-579	839
44	-220	444	149	24	668	558	167	531
45	-338	-234	193	267	-1068	-1026	4	-340
46	-299	203	-27	-236	-240	-579	105	-33
47	629	201	54	26	-183	884	-97	271
48	11	-398	244	-112	47	37	-31	22
49	-428	241	-308	-172	-33	221	16	-12
50	-217	124	-911	477	-109	-872	249	-415
51	-412	-14	52	67	20	-268	-41	-651
52	468	-698	-530	631	34	237	109	-72
53	-446	257	-96	187	-31	38	936	243
54	21	405	836	-76	1043	-517	-444	367
55	-121	593	-53	-212	-1207	-274	979	14
56	97	229	353	1166	417	-724	-315	-170
57	-318	316	211	-26	-188	643	-71	153
58	675	501	-440	-677	378	-10	-158	178
59	249	218	-156	126	-336	308	-768	83
60	-167	-103	235	150	931	-317	1013	132
61	-266	-192	-16	39	549	-1135	283	-30
62	288	822	-42	-38	-55	-27	-5	19
63	49	304	-240	-270	147	463	627	-22
64	28	96	2	-4	-1	-1	1	0
65	-454	89	9	-28	17	1	-5	6
66	137	-214	-255	341	323	-109	-123	-6
67	100	-174	32	-385	-235	-553	408	-680
68	290	-40	24	-43	-350	-279	-305	83
69	-442	-287	102	-90	-85	289	-183	910
70	201	-110	523	-14	-77	27	13	-11
71	37	169	371	-544	-157	115	-738	186
72	437	41	223	619	56	66	-117	63
73	-569	-216	168	375	-201	-671	351	166
74	767	-39	924	-316	-66	295	95	-85
75	-578	-42	200	112	138	-870	597	1784
76	-356	-176	269	-390	654	179	-418	-611
77	-474	-19	47	-29	-9	-581	-560	-88
78	386	289	-191	77	75	8	3	-10

79	-1	-369	-152	-234	73	810	-320	-79
80	-3	-304	-435	147	-287	-97	-56	-12
81	-341	362	-336	402	242	-16	17	8
82	-266	35	-286	-101	-45	-176	-544	544
83	9	421	-12	-139	29	-1278	-870	-642
84	-81	-90	-379	452	80	343	26	-661
85	618	23	-92	-206	84	-219	888	-174
86	-630	-1001	1165	89	-218	-182	-540	174
87	65	-355	-116	-60	-537	757	108	96
88	54	168	-277	465	-1052	1145	-388	-163
89	-1433	210	494	43	-34	118	-160	104
90	163	432	54	224	-2	-582	504	-152
91	450	-295	237	-136	262	-4	-1050	-442
92	53	-137	-132	624	1319	913	-871	-494
93	734	-297	13	237	975	-653	-179	-37
94	-244	942	191	354	-178	95	86	26
95	-821	-155	-173	-14	35	1043	480	-4
96	205	128	209	97	-17	-13	0	-15
97	-380	-184	73	33	509	-108	16	70
98	348	-287	-752	-535	-76	-211	-592	-351
99	-304	162	160	-210	-230	957	177	-851
100	923	-581	68	-828	117	-55	196	181
101	536	-50	391	-4	292	192	681	1085
102	-255	165	923	128	-53	-114	108	110
103	294	-77	15	-2768	398	251	-327	-281
104	243	-255	239	468	-1106	71	476	860
105	-1208	-560	-277	-1	-76	-68	-72	57
106	2123	-978	-144	510	80	-157	-52	-97
107	-89	703	72	-498	37	37	-1921	759
108	488	147	-47	-80	1106	1075	400	-475
109	-237	122	45	162	-1413	-257	-920	560
110	293	877	-490	870	120	95	-504	47
111	55	-11	-257	-31	92	2033	-80	513
112	99	-560	172	-341	-550	-178	67	80
113	-893	486	-266	211	-252	-45	-72	62
114	-216	860	-1981	-912	-24	338	-46	-275
115	115	154	-147	23	447	-163	-123	-1392
116	559	-2138	-119	-473	175	-5	-168	-84
117	103	-162	149	248	-14	-908	1808	-106
118	206	-29	2442	1126	-763	-908	-239	-869
119	-422	-383	-193	415	-1300	899	715	-166
120	458	427	23	2110	-510	-165	251	61
121	-2172	418	-383	-73	-103	138	-18	8
122	1578	1053	451	-261	-208	-280	190	273
123	-356	185	293	184	-116	1399	-1545	13
124	-222	229	559	-1462	2674	219	-155	179
125	346	359	264	214	66	-1777	393	185
126	-2	1063	241	-1342	-487	632	-73	822
127	192	-236	-60	-427	191	399	1107	-1704
128	78	-12	0	0	-1	0	0	0
129	-207	-16	-20	2	0	0	1	0
130	-15	215	-337	-26	-24	-25	-9	-15
131	147	226	282	151	97	79	403	-707
132	461	-451	39	30	-30	15	-31	-19
133	-180	444	12	17	-165	-20	34	880

134	-256	18	349	-34	4	-22	5	-3
135	33	89	-124	-814	-410	194	91	76
136	873	99	-358	26	-33	-97	-8	17
137	-726	79	-67	158	-175	-370	1071	-745
138	411	297	423	-648	-36	-708	46	82
139	476	-76	340	152	-375	-960	-162	922
140	-6	207	-284	-135	710	62	93	-209
141	285	-223	64	74	1	-712	-43	-117
142	-206	537	-50	-9	13	2	-16	-18
143	85	9	-51	264	178	597	24	13
144	7	-375	-108	16	0	3	6	-10
145	-653	573	306	-145	35	-87	-22	16
146	-384	-784	-705	10	14	-20	79	-48
147	783	166	-162	239	-544	-440	-80	-1055
148	359	-211	-277	-369	1597	143	142	201
149	-131	-107	-125	-362	-131	-509	790	446
150	-237	-337	828	906	-103	255	372	-198
151	-300	22	-549	-66	-1183	-5	49	53
152	70	102	-153	834	-542	-184	0	32
153	-582	-375	547	-219	0	14	32	13
154	997	697	0	272	273	85	122	23
155	-483	-100	2	123	175	148	-950	49
156	-721	444	-149	-590	1083	1327	-293	200
157	-241	88	-597	71	1266	-542	-671	327
158	393	1200	742	-199	104	52	-80	-14
159	25	-367	264	-16	45	669	797	266
160	241	-104	-284	-23	-15	0	-2	5
161	-453	-533	-28	-58	-27	-10	-2	-8
162	889	-648	-477	-128	-66	64	-64	-39
163	46	485	-151	-18	185	795	-774	-799
164	793	-1109	494	154	-111	122	73	-25
165	37	-235	-481	-478	46	241	300	1492
166	-71	429	1733	-169	-1464	786	-387	50
167	-579	-522	215	-1508	409	127	837	334
168	970	147	-198	-530	-1197	323	246	-197
169	-1040	214	-126	-36	777	-76	238	-216
170	1593	39	-70	-83	-30	217	-313	-116
171	479	151	-82	-158	254	-734	-977	496
172	-362	-89	514	455	1235	496	259	-125
173	228	-228	104	-622	-1835	-1132	3	-392
174	402	332	-476	-183	-719	-996	-14	224
175	367	-478	24	-959	-78	1530	794	230
176	125	-623	823	-104	237	-223	42	-148
177	-398	893	-421	-490	127	-77	-47	-150
178	198	-701	-2174	422	-216	111	49	-176
179	-552	98	-201	-297	-523	92	-535	-983
180	281	-1599	-751	258	-172	-250	363	257
181	347	178	-459	317	-264	187	1867	664
182	-452	1289	1775	910	895	163	-566	240
183	-485	-293	435	-804	-1326	283	1382	418
184	-590	-283	-540	2410	762	-1001	-858	29
185	-126	-91	586	121	126	1270	-276	-149
186	1558	376	-1173	-112	63	176	120	20
187	1031	88	-409	178	134	683	-2003	-100
188	-1650	31	-26	80	2275	-1033	1070	540

189	113	-695	-129	-642	123	-1933	354	93
190	343	1721	-278	-32	-48	44	-68	-108
191	361	-34	-154	290	61	651	1084	-427
192	105	324	-5	7	-9	24	-3	8
193	-474	-108	-273	202	-49	-9	15	-9
194	307	-99	-899	-28	64	-27	75	24
195	174	19	98	-128	-12	-1608	582	-1432
196	77	-430	-88	54	-647	-244	-1054	-186
197	10	-171	-91	-64	-207	995	-1181	1759
198	217	329	726	287	-310	-320	-624	-94
199	-367	-88	204	-1463	-577	-619	-383	5
200	334	-181	-99	802	487	-176	493	483
201	-951	182	-126	-135	-362	-1769	348	298
202	1475	-56	1636	376	214	10	-129	-115
203	604	291	-488	7	-432	-1293	-690	2563
204	-139	-266	-257	-881	1305	-898	-671	-807
205	-198	-272	-136	285	-36	-1337	-694	299
206	149	610	-715	87	-136	43	90	55
207	211	-643	169	-705	651	1402	-671	230
208	106	-903	-91	-36	101	-72	-53	0
209	-350	98	-653	676	-180	755	-8	515
210	-587	21	-1066	-112	123	14	-10	46
211	440	-30	90	35	-177	-2329	-1985	-737
212	-261	-638	23	635	-56	57	-612	-1173
213	391	383	-25	-646	1193	-1015	1105	210
214	-943	-2115	2321	-29	-122	-33	-286	-13
215	-399	-37	-129	-708	-1108	1132	-816	-205
216	658	-18	382	1570	-211	1670	-285	359
217	-2097	-779	699	373	-22	-187	99	-418
218	842	-146	257	441	-748	-284	926	-118
219	-474	-1	95	185	162	-305	-2023	-381
220	3	58	202	58	2203	253	-1384	433
221	-66	253	250	520	1796	-874	248	-350
222	-789	1663	811	50	-259	-95	83	34
223	-131	505	248	6	-356	1522	1037	73
224	291	194	99	-366	-25	-6	-16	-20
225	-745	-470	137	873	275	-184	-79	442
226	745	-764	-1520	-618	-729	-386	-1621	265
227	408	15	-8	197	14	2043	403	-1155
228	1713	-1318	1191	-1689	1283	-524	-130	45
229	-247	174	159	492	350	1390	773	2264
230	-558	158	1642	-300	85	136	302	-147
231	-441	-524	-1195	-2703	-1644	-48	274	-488
232	-540	-493	496	1682	-2293	-106	901	175
233	-1500	-1580	-790	-327	179	228	-72	-16
234	3818	-1793	795	-227	840	574	-379	77
235	-714	-275	-99	59	-74	-1048	-2344	2005
236	-253	-380	-474	205	1779	1253	1526	-277
237	334	199	67	949	-2780	-982	-539	-78
238	940	1028	-991	1089	1187	-78	746	-272
239	419	71	-99	-144	1177	3674	-643	261
240	-524	-1331	204	-64	-1	-18	32	13
241	-1036	1468	-489	166	49	-134	90	21
242	-919	1301	-2752	838	-750	200	-293	-181
243	-39	-182	61	-66	-143	103	-612	-2924

244	-549	-3477	-351	-24	-487	1011	-165	-127
245	422	330	-126	1231	-965	-1755	2816	457
246	-127	-140	3384	-2027	604	-338	279	-810
247	58	71	169	-473	-2577	1184	-22	234
248	91	-670	949	3684	792	402	311	322
249	-2227	346	-5	1287	-607	2098	-173	299
250	2963	1125	98	505	-110	45	67	-300
251	-40	12	564	-280	-831	835	-3489	-953
252	-181	-863	-166	-46	3603	-817	129	-237
253	-184	171	114	708	802	-3234	-2	-316
254	414	3252	-181	-545	647	353	231	91
255	-12	-23	230	-371	20	549	2529	-709

8.0 Performance of VQ design

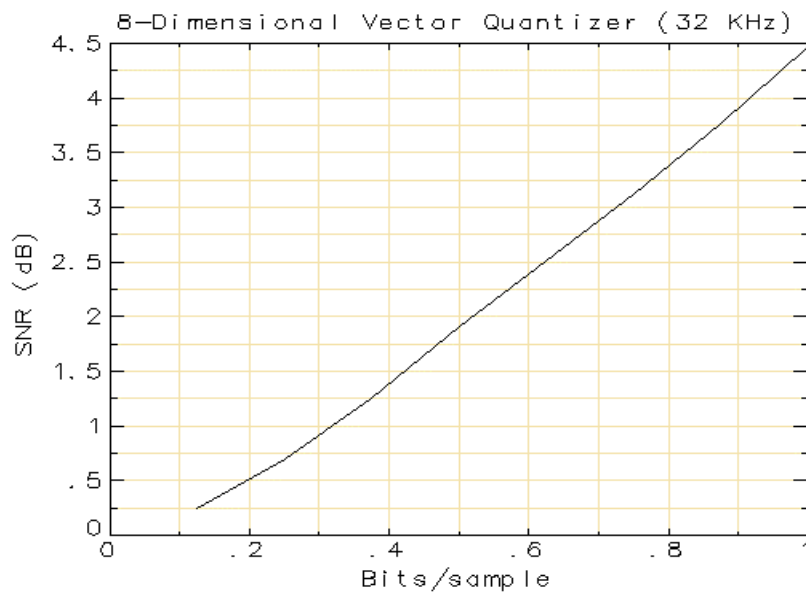


Fig.5 Signal-to-noise characteristics of the 8-dimensional VQ

Fig.5 shows the performance of the VQ as a trade-off between the bit/sample and ensuing distortion introduced by the quantization process. At 1 bit/sample encoding, the distortion introduced in the difference signal is around 4 dB. This was found to be adequate to ensure no audible distortion in the decoded audio sigma stream. Fig.6 shows a profile of the codebook obtained by plotting all the codebook entries.

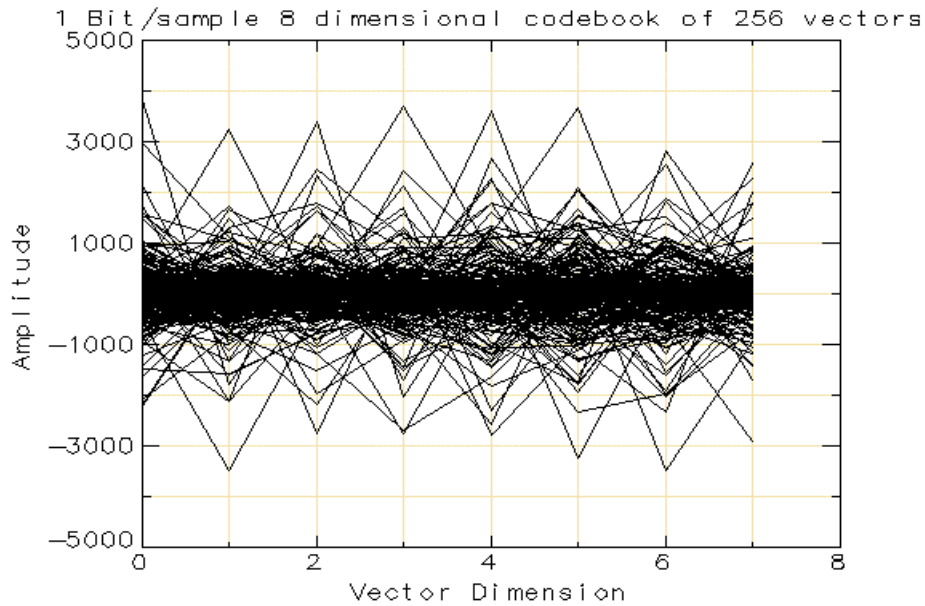


Fig. 6 Waveform Profile of the 8-dimensional VQ

9.0 Program Listings

A list of programs illustrating the concepts is included. They need to be modified to suit overall design methods.

9.1 Vector Quantizer Encoder and Decoder Program

```
/*-----vqtranscode.c -----*/
/*-----V.Ramamoorthy-----*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <io.h>
#include <fcntl.h>
```

```
void print_time(void);
```

```
double out[16];
int yint[256][16], xint[16];
```

```
main (int argc, char **argv)
```

```

{
    long i,j,k,p,l,m,n,nl,dataformat,namax,samax,test,jj,jjj,jvalue;
    double dist,dis,db,xsum,snr,dvalue;
    FILE *fpin, *fpc, *fps, *fpout;
    int fh;
    long pos, filesize, vecnum;
    unsigned char jcode;

    if(argc != 5)
    {
        printf("usage is : vqtranscode codebookfile inputfile codestreamfile decodefile
\n");
        printf(" codebookfile: 256x8 integer codebook file\n");
        printf(" inputfile: output of difintgen routine containing integers\n");
        printf(" codestreamfile: char stream file \n");
        printf(" decodefile: double file to add to subband.dat \n");
        exit(1);
    }

    nl = 256;
    dataformat = 4;

    /*-----Read codebook file -----*/
    if((fpc = fopen(argv[1], "rt")) == NULL)
    {
        printf(" File %s can not be opened for reading! \n", argv[1]);
        exit(1);
    }

    fscanf(fpc, "%ld %ld\n", &k, &n);
    printf("k = %d n = %d -----\n", k, n);
    for(p=0; p < n; p++)
    {
        for(l=0; l < k; l++)
        {
            fscanf(fpc, "%ld %ld %d \n", &i, &j, &yint[p][l]);
            printf("%d %d yvalue = %d \n", p, l, yint[p][l]);
        }
    }
    fscanf(fpc, "%lf\n", &db);
    printf("db = %f\n", db);

    fclose(fpc);

    /*-----*/
    fh = _open(argv[2], _O_RDONLY);
    pos = _lseek(fh, 0L, SEEK_SET);
    if(pos == -1L)
    {
        printf("_lseek to beginning failed\n");
    }
    else
    {
        printf("Position for beginning of file seek = %ld\n", pos);
    }
}

```

```

}

pos = _lseek(fh,0L,SEEK_END);
if(pos == -1L)
{
    printf("_lseek to end failed\n");
}
else
{
    printf("position for end of file seek = %ld\n",pos);
    filesize = pos;
}
_close(fh);

printf("filesize = %d \n",filesize);
samax = filesize/dataformat;

printf("total number of (double)samples = %ld\n",samax);

namax = samax/k;

printf("namax = %d samax = %d k = %d \n",namax,samax,k);

xsum = 0.0;
dis = 0.0;

if((fpin = fopen(argv[2],"rb"))==NULL)
{
    printf(" File %s can not be opened!\n",argv[2]);
    exit(1);
}

if((fps = fopen(argv[3],"wb"))==NULL)
{
    printf(" File %s can not be opened!\n",argv[3]);
    exit(1);
}

if((fpout = fopen(argv[4],"wb"))==NULL)
{
    printf(" File %s can not be opened!\n",argv[4]);
    exit(1);
}

printf(" All files are opened!\n");
print_time();
vecnum = 0;
for(j=0; j < namax; j++)
{
    test = fread(xint,sizeof(int),k,fpin);
    if(test != k)
    {
        printf("error in reading! only %d samples read! \n",test);
        snr = xsum/dis;
        db = 10.0*log10(snr);
    }
}

```

```

        printf("snr = %lf \n",db);
        fclose(fpin);
        fclose(fps);
        fclose(fpout);
        exit(1);
    }

    vecnum++;

    for(jjj=0; jjj < k; jjj++)
    {
        xsum+= xint[jjj]*xint[jjj];
    }

    dvalue = (double)1.0e+30;
    for(jj=0; jj < nl; jj++)
    {
        dist = 0.0;
        for(jjj=0; jjj < k ; jjj++)
        {
            dist += (xint[jjj]-yint[jj][jjj])*(xint[jjj]-yint[jj][jjj]);
        }
        if(dist < dvalue)jvalue = jj;
        if(dist < dvalue)dvalue = dist;
    }
    dis += dvalue;
    jcode = (unsigned char)jvalue;

    fwrite(&jcode, sizeof(char),1,fps);

    for(jjj=0; jjj < k; jjj++)
    {
        out[jjj] = yint[jvalue][jjj]/32767.0;
    }

    fwrite(out,sizeof(double),k,fpout);

}

snr = xsum/dis;
db = 10.0*log10(snr);
printf("xsum = %lf dis = %lf snr = %lf\n",xsum,dis,db);
print_time();

}

#include <time.h>
#include <stdio.h>

struct tm *newtime;
time_t aclock;

void print_time( void )
{

```

```

time( &aclock );          /* Get time in seconds */

newtime = localtime( &aclock ); /* Convert time to struct */
/* tm form */

/* Print local time as a string */
printf( "The current date and time are: %s", asctime( newtime ) );
}

```

9.2 Prediction Difference Generation for Vector Quantization

```

/*---- Difference Generator difintgen.c --- */
/*----- Victor Ramamoorthy ---- 24 Dec 96 --- */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

double r0[32],r1[32];
int out[32];

main(int argc, char** argv)
{
    FILE *fpr0, *fpr1, *fpout;
    int i,j,k,l,m,n,testread,sample,mask,num;

    num = 0;

    if(argc != 4)
    {
        printf(" Usage: difintgen basefile enhancementfile diffile \n");

        exit(1);
    }

    if((fpr0 = fopen(argv[1], "rb")) == NULL)
    {
        printf(" File %s can not be opened for reading! \n",argv[1]);
        exit(1);
    }

    if((fpr1 = fopen(argv[2], "rb")) == NULL)
    {
        printf(" File %s can not be opened for reading! \n",argv[2]);
        exit(1);
    }

    if((fpout = fopen(argv[3], "wb")) == NULL)
    {
        printf(" File %s can not be opened for writing! \n",argv[3]);
    }
}

```

```

        exit(1);
    }

Loop:
    testread = fread(r0, sizeof(double), 32, fpr0);
    if(testread <= 0)
    {
        printf(" Could not read %s : Possibly end of file! \n", argv[1]);
        printf("number of samples written = %d \n",num);
        goto outofloop;
    }
    testread = fread(r1, sizeof(double),32,fpr1);
    if(testread <= 0)
    {
        printf(" Could not read %s : Possibly end of file1 \n",argv[2]);
        printf("number of samples written = %d \n",num);
        goto outofloop;
    }

    for(j=0; j < 32; j++)
    {
        out[j] = (r1[j]-r0[j])*32767.0;
    }

    testread = fwrite(out, sizeof( int),32,fpout);
    if(testread != 32)
    {
        printf(" File %s writing problem! Only %d shorts written! \n",
argv[3],testread);
        exit(1);
    }
    num += 32;
    goto Loop;

outofloop:    fclose(fpr0);
              fclose(fpr1);
              fclose(fpout);

}

```

9.3 Batch File to run Adaptive Media Transcoder

```

cd c:\transcoder\decoder
del *.dat
echo " Action: 1   All dat files in c:\transcoder\decoder are deleted.."

```

```

cd c:\transcoder\base32
del *.dat
echo "Action: 2 All dat files in c:\transcoder\base32 are deleted.."

cd c:\transcoder\enha64
del *.dat
echo " Action: 3 All dat files in c:\transcoder\enha64 are deleted.."

cd c:\transcoder\enha96
del *.dat
echo " Action: 4 All dat files in c:\transcoder\enha96 are deleted.."

cd c:\transcoder\enha128
del *.dat
echo " Action: 5 All dat files in c:\transcoder\enha128 are deleted.."

cd c:\transcoder\diff
del *.dat
echo " Action: 6 All files in c:\transcoder\diff are deleted..."

cd c:\transcoder\mpegfiles
del *.*
echo " Action: 7 All files in c:\transcoder\mpegfiles are deleted..."

cd c:\transcoder\encoder
mpeg3encoder -l 2 -m m -p 2 -s 32.0 -b 32 c:\transcoder\pcmfiles\testmix.pcm
c:\transcoder\mpegfiles\base32.mp2
echo "base 32 mpeg stream is created and stored in c:\transcoder\mpegfiles\base32.mp2
xcopy c:\transcoder\mpegfiles\base32.mp2 c:\transcoder\sigma\

cd c:\transcoder\decoder
mpeg3decoder c:\transcoder\mpegfiles\base32.mp2 c:\transcoder\pcmdecoded\base32dec.pcm
xcopy subband.dat c:\transcoder\base32\
del subband.dat

cd c:\transcoder\encoder
mpeg3encoder -l 2 -m m -p 2 -s 32.0 -b 64 c:\transcoder\pcmfiles\testmix.pcm
c:\transcoder\mpegfiles\enha64.mp2
echo "enha64 mpeg stream is created and stored in c:\transcoder\mpegfiles\enha64.mp2"

cd c:\transcoder\decoder
mpeg3decoder c:\transcoder\mpegfiles\enha64.mp2 c:\transcoder\pcmdecoded\enha64dec.pcm
xcopy subband.dat c:\transcoder\enha64\
del subband.dat

cd c:\transcoder\encoder
mpeg3encoder -l 2 -m m -p 2 -s 32.0 -b 96 c:\transcoder\pcmfiles\testmix.pcm
c:\transcoder\mpegfiles\enha96.mp2
echo "enha96 mpeg stream is created and stored in c:\transcoder\mpegfiles\enha96.mp2

cd c:\transcoder\decoder
mpeg3decoder c:\transcoder\mpegfiles\enha96.mp2 c:\transcoder\pcmdecoded\enha96dec.pcm
xcopy subband.dat c:\transcoder\enha96\
del subband.dat

cd c:\transcoder\encoder

```

```
mpeg3encoder -l 2 -m m -p 2 -s 32.0 -b 128 c:\transcoder\pcmfiles\testmix.pcm
c:\transcoder\mpegfiles\enha128.mp2
echo "enha128 mpeg stream is created and stored in c:\transcoder\mpegfiles\enha128.mp2"
```

```
cd c:\transcoder\decoder
mpeg3decoder c:\transcoder\mpegfiles\enha128.mp2
c:\transcoder\pcmdecoded\enha128dec.pcm
xcopy subband.dat c:\transcoder\enha128\
del subband.dat
echo "-----"
echo " "
echo " "
```

```
cd c:\transcoder\exec
difintgen c:\transcoder\base32\subband.dat c:\transcoder\enha64\subband.dat
c:\transcoder\diff\diff64-32.dat
echo " difintgen is executed...."
```

```
cd c:\transcoder\exec
vqtranscode unicodebook8-256.txt c:\transcoder\diff\diff64-32.dat c:\transcoder\sigma\diff64-32.str c:\transcoder\diff64-32.dec
```

```
cd c:\transcoder\exec
addgen c:\transcoder\base32\subband.dat c:\transcoder\diff64-32.dec
c:\transcoder\decoder\subband.dat
```

```
cd c:\transcoder\decoder
mpeg3decoder c:\transcoder\mpegfiles\enha64.mp2 c:\transcoder\pcmdecoded\sigma64dec.pcm
copy subband.dat c:\transcoder\enha64\subbandnew.dat
```

```
cd c:\transcoder\exec
difintgen c:\transcoder\enha64\subbandnew.dat c:\transcoder\enha96\subband.dat
c:\transcoder\diff\diff96-64.dat
echo " difintgen is executed...."
```

```
cd c:\transcoder\exec
vqtranscode unicodebook8-256.txt c:\transcoder\diff\diff96-64.dat c:\transcoder\sigma\diff96-64.str c:\transcoder\diff96-64.dec
```

```
cd c:\transcoder\exec
addgen c:\transcoder\enha64\subbandnew.dat c:\transcoder\diff96-64.dec
c:\transcoder\decoder\subband.dat
```

```
cd c:\transcoder\decoder
mpeg3decoder c:\transcoder\mpegfiles\enha96.mp2 c:\transcoder\pcmdecoded\sigma96dec.pcm
copy subband.dat c:\transcoder\enha96\subbandnew.dat
```

```
cd c:\transcoder\exec
difintgen c:\transcoder\enha96\subbandnew.dat c:\transcoder\enha128\subband.dat
c:\transcoder\diff\diff128-96.dat
echo " difintgen is executed...."
```

```
cd c:\transcoder\exec
```

```
vtqtranscode unicodebook8-256.txt c:\transcoder\diff\diff128-96.dat c:\transcoder\sigma\diff128-96.str c:\transcoder\diff128-96.dec
```

```
cd c:\transcoder\exec  
addgen c:\transcoder\enha96\subbandnew.dat c:\transcoder\diff128-96.dec  
c:\transcoder\decoder\subband.dat
```

```
cd c:\transcoder\decoder  
mpeg3decoder c:\transcoder\mpegfiles\enha128.mp2  
c:\transcoder\pcmdecoded\sigma128dec.pcm  
copy subband.dat c:\transcoder\enha128\subbandnew.dat
```

10.0 References

- [1] J.D. Johnston, "Transform Coding of Audio Signals Using Perceptual Noise Criteria", IEEE Journal on Selected Areas in Communications, February 1988, vol.6, no.2, pp. 314-323.
- [2] Davis Pan, "A Tutorial on MPEG/Audio Compression", IEEE Multimedia Journal, Summer 1995, pp. 60-74.
- [3] Robert M. Gray, "Vector Quantization", IEEE ASSP Magazine, April 1984, pp. 4-29.
- [4] Ronald E. Crochiere and Lawrence R. Rabiner, "Multirate Digital Signal Processing", Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 1983. Chapter 4.1 to 4.2.5.