

Optimized Recursive Subband Synthesis Windowing for Implementing Efficient MPEG Audio Decoders

Patrick De Smet, *Student Member, IEEE*, and Ignace Bruyland, *Member, IEEE*

Abstract—In this letter, we discuss a recursive optimization approach for the subband synthesis windowing component used in all three layers of the MPEG-1 and -2 audio decoding frameworks. We explain the design of the proposed algorithm and briefly discuss how it can be combined with other novel and existing optimization strategies. The impact of these optimizations on the number of multiplications and additions is shown, and the influence of the algorithm on the amount of memory required by the adapted decoder is also discussed. We show that the proposed algorithm offers very interesting possibilities for implementing efficient VLSI designs.

Index Terms—Efficient decoder, MPEG audio, recursive windowing.

I. INTRODUCTION

IN THE LAST ten years, the ISO/IEC MPEG-1 and -2 audio encoding standards have been gaining considerable importance and popularity as efficient tools for lossy compression of high-quality digital audio [1], [2]. This letter aims at contributing other, previously undocumented possibilities for obtaining faster and/or more cost-effective MPEG audio decoders. We mainly focus on realizing more efficient VLSI designs by reducing the number of multiplications; compared to adder components, multipliers are relatively slow and often result in considerably more demanding design restrictions [3]. The proposed algorithm is based upon the optimization of redundant computations carried out in subsequent synthesis filtering operations and also extends and complements techniques discussed in [4]–[6] (encoding only), [7]–[9] (decoding only), and [10]–[12] (both).

II. MPEG AUDIO DECODING AND SUBBAND SYNTHESIS

Within the MPEG-1 and -2 audio encoding standards, ISO/IEC has defined three different “layers” [1], [13]. Each layer builds upon its lower ranked siblings, and for each higher layer number, (implementation) complexity and compression efficiency typically both increase. The cosine-modulated polyphase filter bank used by all three layers within the framework maps the digital audio input into 32 equally spaced and critically decimated bandpass signals [2]. Compared to the encoder, the computational complexity of the MPEG audio

decoder is significantly lower; most of the computation time is due to the synthesis filter bank [7], [9].

For each multirate filtering step at time t , the ISO synthesis subband filtering algorithm starts with $M = 32$ subband samples extracted from the encoded bitstream syntax, i.e., $S_{sb}[i; t]$, $i = 0, 1, \dots, M - 1$, and produces M pulse code modulation (PCM) audio output samples; $S_{pcm}[j; t]$, $j = 0, 1, \dots, M - 1$.

First, the intermediate inverse polyphase quadrature mirror filter bank (IPQMF) vector $V[i; t]$ is shifted over $2M$ memory positions according to $V[i; t] = V[i - 2M; t - 1]$, for $i = 2PM - 1, 2PM - 2, \dots, 2M$ with $P = N/M = 16$ and $N = 512$ the length of the prototype filter window [2]. Then, the basic MPEG audio matrixing operation [13] is calculated

$$V[i; t] = \sum_{k=0}^{M-1} S_{sb}[k; t] \cos \left(\frac{\pi(2k+1) \left(\frac{M}{2} + i \right)}{2M} \right) \quad (1)$$

for $i=0, 1, \dots, 2M-1$. An N -value intermediate vector $U[i; t]$ is then built using $U[2Mk+l; t] = V[4Mk+l; t]$, and $U[2Mk+M+l; t] = V[4Mk+3M+l; t]$, $k = 0, 1, \dots, P/2-1$, $l = 0, 1, \dots, M-1$. Finally, the so-called windowing component computes M new PCM audio output samples, i.e., $S_{pcm}[j; t] = \sum_{i=0}^{P-1} U[j+Mi; t] D[j+Mi]$, $j=0, 1, \dots, M-1$, in which the values D are the N synthesis window coefficients (see [13, Table B.3]).

This algorithm requires $2M^2 + MP = 2560$ multiplications and $2M(M-1) + M(P-1) = 2464$ additions, and, not counting the PCM and subband sample buffers, at least $3N + M = 1568$ datawords of memory; $2PM = 2N = 1024$ datawords V , $M = 32$ cosine-coefficients (contrary to [15], using all 2048 cosine-coefficients in (1), we assume appropriate address calculators always being implemented), and $N = 512$ window coefficients D .

III. MATRIXING OPTIMIZATIONS

Obviously, (1) can be simplified based on the properties of the cosine-function. This has been discussed in [7], [9], and [12], yielding a fast odd/even butterfly M -point inverse discrete cosine transform decomposition (see also [14]). However, except for the fact that, for $l = 0, 1, \dots, M/2 - 1$

$$\begin{aligned} \text{SDS}_0[l; t] &= S_{sb}[l; t] + S_{sb}[M-1-l; t] \\ \text{SDS}_1[l; t] &= S_{sb}[l; t] - S_{sb}[M-1-l; t] \\ V[i; t] &= \sum_{k=0}^{M/2-1} \text{SDS}_{(i \bmod 2)}[k; t] \cos \left(\frac{\pi(2k+1) \left(\frac{M}{2} + i \right)}{2M} \right) \end{aligned}$$

Manuscript received December 18, 2001; revised November 15, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Antonio Ortega.

The authors are with the Department of Telecommunications and Information Processing (TW07), Ghent University, B-9000 Ghent, Belgium (e-mail: pds@telin.UGent.be; ib@telin.UGent.be).

Digital Object Identifier 10.1109/LSP.2003.817173

with

$$\begin{cases} V[i; t] = -V[M - i; t], & i = 0, 1, \dots, \frac{M}{2} - 1 \\ V[M + i; t] = V[2M - i; t], & i = 1, 2, \dots, \frac{M}{2} - 1 \end{cases}$$

$$V[M/2; t] = 0 \quad V[3M/2; t] = - \sum_{k=0}^{M/2-1} \text{SDS}_0[k; t] \quad (2)$$

we do not repeat this discussion here; we will reduce the computational complexity by pursuing other possibilities. The values for sum/difference-symmetry (SDS) reflect the symmetry due to the factor $(2k + 1)$ in the cosine argument of (1).

The partially optimized matrixing formulas given above require only $(M - 1)M/2 = 496$ multiplications and $M + (M/2 - 1)(M - 1) + (M/2 - 1) = M^2/2 = 512$ additions to compute the $V[i; t]$ matrixing results. If needed, these equations can also reduce the required amount of memory for the entire filtering procedure to $2N + 2M = 1088$ datawords; $M = 32$ values SDS_0 and SDS_1 , $PM = N = 512$ remaining values V , $M = 32$ cosine-coefficients, and $N = 512$ window coefficients D .

IV. RECURSIVE WINDOWING OPTIMIZATIONS

For VLSI design, multiplications are considered significantly more expensive than additions; the required multiplier speed can often influence the choice of the entire implementation design (see the *encoder* architecture trade-offs discussed in [3] and [4]). Hence, in this section we focus on reducing the number of multiplications in the windowing component.

Now, since

$$S_{\text{pcm}}[j; t] = \sum_{i=0}^{P/2-1} (U[j + Mi; t] D[j + Mi]) + \left(D \left[j + Mi + \frac{MP}{2} \right] U \left[j + Mi + \frac{MP}{2}; t \right] \right)$$

and $ab + cd = (a + c)(b + d) - ad - bc$, we have

$$\begin{aligned} S_{\text{pcm}}[j; t] &= \sum_{i=0}^{P/2-1} \left(U[j + Mi; t] + D \left[j + Mi + \frac{MP}{2} \right] \right) \\ &\quad \cdot \left(D[j + Mi] + U \left[j + Mi + \frac{MP}{2}; t \right] \right) \\ &\quad - \sum_{i=0}^{P/2-1} U[j + Mi; t] U \left[j + Mi + \frac{MP}{2}; t \right] \\ &\quad - \sum_{i=0}^{P/2-1} D[j + Mi] D \left[j + Mi + \frac{MP}{2} \right] \quad (3) \end{aligned}$$

for $j = 0, 1, \dots, M - 1$. Rewriting this sum as $S_{\text{pcm}}[j; t] = A[j; t] - B[j; t] - C[j]$, the last sum $C[j]$ is an M -dimensional constant vector which can be precalculated for each j . Below, each of these terms will be discussed in more detail, but first let

us consider some more general issues. The A terms will remain to be calculated by the final algorithm, but the B terms can be simplified further by splitting them into two sums for even and odd multiples of M for the index i , and by replacing U by V (see Section II). This yields

$$B[j; t] = \sum_{i=0}^{P/4-1} (V[j + 4Mi; t] V[j + 4Mi + MP; t] + V[j + 4Mi + 3M; t] V[j + 4Mi + MP + 3M; t]) \quad (4)$$

As already indicated, t is an external clocking signal driving the subband synthesis filter; for each filter calculation at time t , $2M$ new values $V[l; t]$ are determined according to (1) or (2). Considering the previous calculations of the subband synthesis filter, it is clear that $V[l + 2Mn; t] = V[l; t - n]$, for $l = 0, 1, \dots, (2PM - 1 - 2Mn)$, or equivalently, $V[l; t] = V[l - 2Mn; t - n]$, $l = (2Mn), \dots, 2PM - 2, 2PM - 1$. Hence

$$B[j; t] = B[j; (t - 2)] - d[j; (t - 8)] + d[j; t]$$

with

$$d[j; t] = V[j; t] V[j + MP; t] + V[j + 3M; t] V[j + 3M + MP; t] \quad (5)$$

for $j = 0, 1, \dots, M - 1$. This result can be easily obtained from (4) by removing the last terms for $i = P/4 - 1 = 3$ and adding new terms for $i = 0$, and shifting over two clocking periods t or $4M = 128$ positions.

The equations in this section yield the proposed recursive algorithm for calculating the windowing component. To our knowledge this decoder structure has not been proposed before. The recursive computation of $B[j; t]$ allows the number of multiplications to be reduced by recycling data computed in previous filtering operations [using two separate delay lines for B and d ; see (5)]. Below, we will discuss the memory requirements and the computational complexity of the proposed algorithm, but first let us discuss some additional possibilities for further optimization.

V. ADDITIONAL OPTIMIZATIONS

A. Detailed Optimizations for the Recursive Algorithm

From the equations given above, it is clear that $V[M/2; t] = V[M/2 + MP; t] = 0$; and hence, $d[M/2; t] = V[M/2 + 3M; t] V[M/2 + MP + 3M; t]$, using only one multiplication to compute $d[M/2; t]$. The value $d[0; t]$ can be calculated by using two multiplications and one addition. Finally, based on the properties of V [see (2)] and the definition of d given above, it can be easily shown that $d[j; t] = d[M - j; t]$, $j = 1, 2, \dots, M/2 - 1$. Hence, the number of multiplications and additions for calculating all $d[j; t]$ can be halved. Additionally, this also means that $B[j; t] = B[M - j; t]$, $j = 1, 2, \dots, M/2 - 1$, and that $B[0; t]$ and $B[M/2; t]$ are unique.

B. Windowing Symmetry

Another possibility for further optimization is given by the fact that $\{D[0]=0\}$

$$\left\{ D\left[\frac{N}{2}+k\right] = -D\left[\frac{N}{2}-k\right] \right. \\ \left. \text{for } k \neq 2M, 4M, 6M, k=1, 2, \dots, \frac{N}{2}-1 \right\}$$

and $\{D[k] = D[N-k]; k=2M, 4M, 6M\}$, and $D[N/2]$ are unique. To our knowledge, these properties have not received any real attention in the existing literature. This seems odd, since clearly, exploiting this symmetry will immediately yield a significant advantage in the amount of memory needed to store these coefficients. Also, for $j = 1, 2, \dots, M/2 - 1$, from (3) and with $PM = N$ (see above), we get (6) (shown at the bottom of the page). Assuming now that $i=P/2-1-l$ in each parallel computation step for a fixed $j=1, 2, \dots, M/2-1$, and exploiting the symmetry of the window coefficients, i.e., $D[j+Mi] = -D[M-j+Mi+N/2]$ and $-D[M-j+Mi] = D[j+Mi+N/2]$, half of the window D suffices to compute all values A . Also, note that the parallel computations in (6) can be used to avoid reloading the coefficients D (from external memory) into internal registers. Next, by considering that $D[0] = 0$ and $U[M/2+2Mn;t] = V[M/2+4Mn;t] = V[M/2;t-2n] = 0$; it is clear that

$$A[0;t] = \left(U[0;t] + D\left[\frac{N}{2}\right] \right) U\left[\frac{N}{2};t\right] \\ + \sum_{i=1}^{P/2-1} \left(U[Mi;t] + (-1)^i \right. \\ \left. \cdot D\left[\frac{N}{2} - Mi\right] \right) \left(D[Mi] + U\left[Mi + \frac{N}{2};t\right] \right)$$

and

$$A\left[\frac{M}{2};t\right] = A_{M/2} + \sum_{i=0}^{P/4-1} \left(U\left[\frac{M}{2} + M + 2Mi;t\right] \right. \\ \left. - D\left[\frac{N}{2} - \frac{M}{2} - M - 2Mi\right] \right) \\ \cdot \left(D\left[\frac{M}{2} + M + 2Mi\right] \right. \\ \left. + U\left[\frac{N}{2} + M + \frac{M}{2} + 2Mi;t\right] \right)$$

with the constant $A_{M/2} = -\sum_{i=0}^{P/4-1} D[N/2 - M/2 - 2Mi] D[M/2 + 2Mi]$. Finally, for the constants C it can be shown that $C[j] = C[M-j] = \sum_{i=0}^{P/2-1} D[j+Mi] (-D[N/2-j-Mi])$, for $j=1, 2, \dots, M/2-1$. Additionally, $C[M/2] = \sum_{i=0}^{P/2-1} D[M/2+Mi] (-D[N-M/2-Mi])$, and $C[0] = (D[MP/4])^2 + 2 \sum_{i=1}^{P/4-1} (-1)^i D[Mi] D[N/2-Mi]$. These equations can aid in a faster (re)start of a decoder when the C -values are not hard-coded into the actual application or the ROM of a device.

VI. FINAL OPTIMIZED ALGORITHM

Initially, the constants $A_{M/2}$, $C[0]$, $C[M/2]$, and $C[j]$, $j=1, 2, \dots, M/2-1$, need to be available. All other values are initialized to zero but both $B[j;t=0]$ and $B[j;t=1]$ are set to $C[j]$ for $j=0, 1, \dots, M/2$. This absorbs the values C into the recursive B s, simplifying the computation of the PCM output samples $S_{pcm}[j;t]$. After extracting the M encoded subband samples $S_{sb}[j;t]$ from the bitstream, the matrixing operations are computed by using (2). Next, $d[j;t]$ is computed according to the definition given in Section IV and the symmetry issues discussed in Section V-A. This enables the computation of all $B[j;t]$, again according to Sections V-A and IV. Then, in a single loop, $A[j;t]$ and $A[M-j;t]$, and next, $A[0;t]$ and $A[M/2;t]$ can be computed using the equations formulated above. Finally, we can compute $S_{pcm}[j;t] = A[j;t] - B[j;t]$ for outputting samples.

Adapted source code for the ISO codec [15] is freely available from the authors.¹

VII. COMPUTATIONAL COMPLEXITY

A. Recursive Synthesis Windowing

The original windowing component as described in Section II requires $MP=512$ multiplications and $M(P-1)=480$ additions for calculation of the PCM samples.

In the proposed recursive algorithm, the computation of the values $A[j;t]$, requires $(M-2)P/2 = 240$ multiplications and $2(M-2)(P/2) + (M-2)(P/2-1) = 690$ additions for $j=1, 2, \dots, M/2-1, M/2+1, M/2+2, \dots, M-1$; see (6). The computation of $A[0;t]$ requires $1 + (P/2-1) = 8$ multiplications and $1 + 2(P/2-1) + (P/2-1) = 22$ additions. For the calculation of $A[M/2;t]$ we need $P/4 = 4$ multiplications and $2(P/4) + P/4 = 12$ additions. This makes a total of 252 multiplications and 724 additions. The computation of all M values d requires $1 + 2 + 2(M/2-1) = 33$ multiplications and $1 + (M/2-1) = 16$ additions. Similarly, exploiting the

¹See also <http://telin.UGent.be/~pds/papers/SPL2003/>.

$$A[j;t] = \sum_{i=0}^{P/2-1} \left(U[j+Mi;t] + D\left[j+Mi + \frac{N}{2}\right] \right) \left(D[j+Mi] + U\left[j+Mi + \frac{N}{2};t\right] \right) \\ A[M-j;t] = \sum_{i=P/2-1}^0 \left(U[M-j+Mi;t] + D\left[M-j+Mi + \frac{N}{2}\right] \right) \left(D[M-j+Mi] + U\left[M-j+Mi + \frac{N}{2};t\right] \right) \quad (6)$$

symmetry in B (see also Section V-A), only $2(M/2 + 1) = 34$ additions are needed for the computation of all values B . Finally, for the calculation of the PCM output samples, adding A and B (with C absorbed into B) requires $M = 32$ additions. Hence, 285 multiplications and 806 additions are needed to implement the optimized recursive windowing algorithm. The proposed algorithm results in an increase in the required number of additions. However, the number of multiplications has been reduced significantly; as already mentioned, this will normally put considerably less restricting demands on (the speed of) a VLSI or hardware architecture [3].

B. Full Subband Synthesis Filtering

In summary, when compared to the full basic ISO subband synthesis filter as discussed in Section II, the computational complexity is reduced from 2560 multiplications and 2464 additions (see Section II), to 781 multiplications and 1318 additions (see Section III: 496 multiplications and 512 additions; and, Section VII-A: 285 multiplications and 806 additions).

VIII. MEMORY REQUIREMENTS

A. Recursive Synthesis Windowing

Implementing the proposed algorithm requires $8(1 + M/2) = 136$ datawords for $1 + M/2 = 17$ values $d[j; t]$ with a maximum time delay of eight synthesis operations, $2(1 + M/2) = 34$ datawords for $B[j; t]$ due to a buffer of 2 filtering operation delays, one dataword for $A_{M/2}$, and the $1 + M/2 = 17$ constants C . However, as indicated above, these constants C are absorbed into the recursive computation of B , and they can be easily computed when the decoder is being initialized; the constants C are not needed during "steady state" operation of the decoder. Thus, the recursive algorithm can be implemented with a total of $10(1 + M/2) + 1 = 5M + 11 = 171$ additional datawords. Fortunately, by also exploiting the symmetry in the coefficients D (see Section V-B), we can reclaim $N/2 = 256$ datawords. Hence, compared to the traditional implementations described in the existing literature, the overall balance of all the optimizations we proposed in this letter is a gain of $N/2 - (5M + 11) = 85$ datawords.

B. Full Subband Synthesis Filtering

In this letter, we discussed and proposed the memory-related optimization of the MPEG matrixing operations (see Section III), the auxiliary recursive data (see Section V-A), and the window coefficients (see Section V-B). After implementing all these optimizations, the required amount of memory can be reduced from $3N + M = 1568$ datawords (see Section II) to $(2N + 2M) - (N/2 - 5M - 11) = 1003$ datawords; see Sections III and VIII-A.

Note that not all symmetry properties must be implemented. This results in several flexible trade-off possibilities for ex-

changing address calculation logic and control complexity for memory requirements and vice versa.

IX. CONCLUSION

In this letter, we proposed a recursive windowing algorithm useful for implementing efficient MPEG-1 and -2 audio subband synthesis filters. The proposed windowing algorithm reduces the required number of multiplications from 512 to 285, while the number of additions changes from 480 to 806. This approach can be very profitable when a certain hardware design has some limitations on (the speed of) its multiplier(s). Additionally, we proposed and discussed several memory related requirements and optimizations. In combination with an existing and well-known cosine-symmetry exploiting optimization step, this letter simplifies the original ISO subband synthesis filtering procedure from 2560 multiplications and 2464 additions to 781 multiplications and 1318 additions, while reducing the required amount of memory from 1568 to 1003 datawords.

REFERENCES

- [1] P. Noll, "MPEG digital audio coding," *IEEE Signal Processing Mag.*, pp. 59–81, Sept. 1997.
- [2] M. Kahrns and K. Brandenburg, Eds., *Applications of Digital Signal Processing to Audio and Acoustics*. Dordrecht, The Netherlands: Kluwer, 1998.
- [3] N. Ling and C.-W. Shih, "Two polyphase filter architectures for MPEG audio," in *Proc. 40th Midwest Symp. Circuits and Systems*, vol. 2, 1997, pp. 861–864.
- [4] D.-Y. Chan, J.-F. Yang, and C.-C. Fang, "Fast implementation of MPEG audio coder using recursive formula with fast discrete cosine transforms," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 144–148, Mar. 1996.
- [5] M. Kumar and M. Zubair, "A high performance software implementation of MPEG audio encoder," *Proc. ICASSP*, vol. 2, pp. 1049–1052, 1996.
- [6] H.-O. Oh, S.-Y. Kim, D.-H. Youn, and I.-W. Cha, "New implementation techniques of a real-time MPEG-2 audio encoding system," *Proc. ICASSP*, vol. 4, pp. 2287–2290, 1999.
- [7] T.-H. Tsai, T.-H. Chen, and L.-G. Chen, "An MPEG audio decoder chip," *IEEE Trans. Consumer Electron.*, vol. 41, pp. 89–96, Feb. 1995.
- [8] P. De Smet, "On two new techniques for optimized MPEG audio decoding," in *Proc. IASTED Int. Conf. SPPRA 2001*, July 3–6, 2001, pp. 79–83.
- [9] W.-K. Paik and S.-Y. Hwang, "Design of a novel synthesis filter for real-time MPEG-2 audio decoder implementation on a DSP chip," *IEEE Trans. Consumer Electron.*, vol. 45, pp. 1119–1129, Nov. 1999.
- [10] C. D. Murphy and L. Anandakumar, "Real-time MPEG-1 audio coding and decoding on a DSP chip," *IEEE Trans. Consumer Electron.*, vol. 43, pp. 40–47, Feb. 1997.
- [11] V. Britanak and K. R. Rao, "An efficient implementation of the forward and inverse MDCT in MPEG audio coding," *IEEE Signal Processing Lett.*, vol. 8, pp. 48–51, Feb. 2001.
- [12] K. Konstantinides, "Fast subband filtering in MPEG audio coding," *IEEE Signal Processing Lett.*, vol. 1, pp. 26–28, Feb. 1994.
- [13] ISO/IEC, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s," ISO/IEC 11172-3, 1993.
- [14] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1243–1245, Dec. 1984.
- [15] ISO dist10 Source Code Package [Online]. Available: /mpeg2/software/at ftp://ftp.tnt.uni-hannover.de/pub/MPEG/audio/.