DCA Transform Deobfuscated

Alexander E. Patrakov

1 Official definition

The transform that takes the subband samples and produces PCM samples is officially specified using pseudocode [1]. When translated to C, this yields:

```
double *prCoeff;
double dct4_coeff[16][16], dct2_coeff[16][16];
double COS_table[16], SIN_table[16];

void
PreCalArrays (void)
{
  int i, k;
  for (k = 0; k < 16; k++)
    for (i = 0; i < 16; i++)
      dct4_coeff[k][i] = cos ((2 * i + 1) * (2 * k + 1) * M_PI / 64);
  for (k = 0; k < 16; k++)
    for (i = 0; i < 16; i++)
      dct2_coeff[k][i] = cos ((i) * (2 * k + 1) * M_PI / 32);
  for (k = 0; k < 16; k++)
    COS_table[k] = 0.25 / (2 * cos ((2 * k + 1) * M_PI / 128));
  for (k = 0; k < 16; k++)
    SIN_table[k] = -0.25 / (2 * sin ((2 * k + 1) * M_PI / 128));
}

/* raXin contains 32 subband samples for a given channel,
   raXout receives 32 PCM samples for this channel */

void
QMFInterpolation2 (double *raXin, double *raXout)
{
  int i, j, k;
  double A[16], B[16], SUM[16], DIFF[16];

  static double raX[512];
  static double raZ[64];

  /* raXin contains 32 subband samples for a given channel */

  for (k = 0; k < 16; k++)
    {
      A[k] = 0.0;
      for (i = 0; i < 16; i++)
        A[k] += (raXin[2 * i] + raXin[2 * i + 1]) * dct4_coeff[k][i];
    }

  for (k = 0; k < 16; k++)
    {
      B[k] = 0.0;
      for (i = 0; i < 16; i++)
        {
          if (i > 0)
            B[k] += (raXin[2 * i] + raXin[2 * i - 1]) * dct2_coeff[k][i];
          else
            B[k] += (raXin[2 * i]) * dct2_coeff[k][i];
        }
      SUM[k] = A[k] + B[k];
      DIFF[k] = A[k] - B[k];
    }

  for (k = 0; k < 16; k++)
    raX[k] = COS_table[k] * SUM[k];
  for (k = 0; k < 16; k++)
    raX[32 - k - 1] = SIN_table[k] * DIFF[k];

  for (k = 31, i = 0; i < 32; i++, k--)
    {
      for (j = 0; j < 512; j += 64)
        raZ[i] += prCoeff[i + j] * (raX[i + j] - raX[j + k]);
    }

  for (k = 31, i = 0; i < 32; i++, k--)
    for (j = 0; j < 512; j += 64)
      raZ[32 + i] += prCoeff[32 + i + j] * (-raX[i + j] - raX[j + k]);

  for (i = 0; i < 32; i++)
    raXout[i] = raZ[i];

  for (i = 511; i >= 32; i--)
    raX[i] = raX[i - 32];
  for (i = 0; i < 32; i++)
    raZ[i] = raZ[i + 32];
  for (i = 0; i < 32; i++)
    raZ[i + 32] = 0.0;
}
```

As you can see, the code is obfuscated. The aim of this paper is to document the steps needed to deobfuscate it.

2 From C to math

Let's translate the code above to mathematical definitions. The relation between notations used in the code and in the math formulae is shown below. In this table, i takes the values $0 \ldots 31$ and enumerates the subbands.

Code	Formulae
`raXin[i]`	$x_{0,i}$
`raXin[i]` j invocations ago	$x_{-j,i}$
`A[k]`, `B[k]`	A_k, B_k
`SUM[k]`, `DIFF[k]`	S_k, D_k
`raX[k]`	X_k
`raZ[k]`	Z_k
`prCoeff[k]`	C_k

Since the code implements some linear transformation of its inputs and their history, and operates by portions of 32 values, it is reasonable to expect that its outputs Z_k can be related to its inputs with the following relation:

$$Z_k = \sum_{j'} \sum_{i=0}^{31} K_{32j'+k,i} x_{i,-j'}, \tag{1}$$

i.e. is a sum of 32 convolutions of kernels $K_{\ldots,i}$ (to be found) with the 32 input signals upsampled by a factor of 32. At the end, we'll see that it is indeed the case.

The code assigns the following values to temporary arrays:

$$A_k = \sum_{i=0}^{15} (x_{0,2i} + x_{0,2i+1}) \cos \frac{(2i+1)(2k+1)\pi}{64}, \quad k = 0 \ldots 15 \tag{2}$$

$$B_k = \sum_{i=0}^{15} (x_{0,2i} + x_{0,2i-1}) \cos \frac{(2i)(2k+1)\pi}{64}, \quad k = 0 \ldots 15 \tag{3}$$

where it is assumed that $x_{0,-1} = 0$.

Then, the following quantities are stored in the beginning of the `raX[]` array:

$$X_k = \frac{A_k + B_k}{8 \cos \frac{(2k+1)\pi}{128}}, \quad k = 0 \ldots 15 \tag{4}$$

$$X_{31-k} = -\frac{A_k - B_k}{8 \sin \frac{(2k+1)\pi}{128}}, \quad k = 0 \ldots 15 \tag{5}$$

These expressions can be simplified. First, let's consider $A_k + B_k$, expand it and thus find factors that apply to $x_{0,i}$ for each i:

$$S_k = A_k + B_k = S_k^{(e)} + S_k^{(o)} \tag{6}$$

where

$$S_k^{(e)} = \sum_{i=0}^{15} x_{0,2i} \left(\cos \frac{(2i+1)(2k+1)\pi}{64} + \cos \frac{(2i)(2k+1)\pi}{64} \right) \tag{7}$$

$$S_k^{(o)} = \sum_{i=0}^{15} \left(x_{0,2i+1} \cos \frac{(2i+1)(2k+1)\pi}{64} + x_{0,2i-1} \cos \frac{(2i)(2k+1)\pi}{64} \right) \tag{8}$$

Let's simplify $S_k^{(e)}$ by using the sum-of-cosines identity:

$$\cos\alpha + \cos\beta = 2 \cos \frac{\alpha+\beta}{2} \cos \frac{\alpha-\beta}{2} \tag{9}$$

This yields:

$$S_k^{(e)} = 2\cos\frac{(2k+1)\pi}{128}\sum_{i=0}^{15} x_{0,2i}\cos\frac{(4i+1)(2k+1)\pi}{128} \tag{10}$$

The expression for $S_k^{(o)}$ can be simplified, if one goes from i to $i' = i-1$ in the second term (i.e., puts $i = i'+1$), shifting the summation limits accordingly:

$$S_k^{(o)} = \sum_{i=0}^{15} x_{0,2i+1}\cos\frac{(2i+1)(2k+1)\pi}{64} + \sum_{i'=-1}^{14} x_{0,2i'+1}\cos\frac{(2i'+2)(2k+1)\pi}{64} \tag{11}$$

Note that in the second sum, we can drop the term with $i' = -1$ because $x_{0,-1} = 0$, and add the term with $i' = 15$ because in this case the cosine argument will be an odd multiple of $\pi/2$. In other words, the dropped and added terms are always zero. Therefore, we can write the two sums as one and obtain:

$$S_k^{(o)} = \sum_{i=0}^{15} x_{0,2i+1}\left(\cos\frac{(2i+1)(2k+1)\pi}{64} + \cos\frac{(2i+2)(2k+1)\pi}{64}\right) \tag{12}$$

Using the sum-of-cosines identity again, we find:

$$S_k^{(o)} = 2\cos\frac{(2k+1)\pi}{128}\sum_{i=0}^{15} x_{0,2i+1}\cos\frac{(4i+3)(2k+1)\pi}{128} \tag{13}$$

Thus, by summing (10) and (13), we obtain:

$$A_k + B_k = 2\cos\frac{(2k+1)\pi}{128}\sum_{i=0}^{31} x_{0,i}\cos\frac{(2i+1)(2k+1)\pi}{128} \tag{14}$$

Then, we have to consider $A_k - B_k$:

$$D_k = A_k - B_k = D_k^{(e)} + D_k^{(o)} \tag{15}$$

where

$$D_k^{(e)} = \sum_{i=0}^{15} x_{0,2i}\left(\cos\frac{(2i+1)(2k+1)\pi}{64} - \cos\frac{(2i)(2k+1)\pi}{64}\right) \tag{16}$$

$$D_k^{(o)} = \sum_{i=0}^{15}\left(x_{0,2i+1}\cos\frac{(2i+1)(2k+1)\pi}{64} - x_{0,2i-1}\cos\frac{(2i)(2k+1)\pi}{64}\right) \tag{17}$$

Let's simplify $D_k^{(e)}$ by using the difference-of-cosines identity:

$$\cos\alpha - \cos\beta = -2\sin\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2} \tag{18}$$

This yields:

$$D_k^{(e)} = -2\sin\frac{(2k+1)\pi}{128}\sum_{i=0}^{15} x_{0,2i}\sin\frac{(4i+1)(2k+1)\pi}{128} \tag{19}$$

The expression for $D_k^{(o)}$ can be simplified, as before, by putting puts $i = i' + 1$ in the second term:

$$D_k^{(o)} = \sum_{i=0}^{15} x_{0,2i+1} \cos \frac{(2i+1)(2k+1)\pi}{64} - \sum_{i'=-1}^{14} x_{0,2i'+1} \cos \frac{(2i'+2)(2k+1)\pi}{64} \tag{20}$$

As before, in the second sum, we can drop the term with $i' = -1$ and add the term with $i' = 15$ without changing the sum. Therefore, we can write the two sums as one and obtain:

$$D_k^{(o)} = \sum_{i=0}^{15} x_{0,2i+1} \left(\cos \frac{(2i+1)(2k+1)\pi}{64} - \cos \frac{(2i+2)(2k+1)\pi}{64} \right) \tag{21}$$

Using the difference-of-cosines identity again, we find:

$$D_k^{(o)} = 2 \sin \frac{(2k+1)\pi}{128} \sum_{i=0}^{15} x_{0,2i+1} \sin \frac{(4i+3)(2k+1)\pi}{128} \tag{22}$$

Thus, by summing (10) and (13), we obtain:

$$A_k - B_k \quad = \quad -2 \sin \frac{(2k+1)\pi}{128} \sum_{i=0}^{31} (-1)^i x_{0,i} \sin \frac{(2i+1)(2k+1)\pi}{128} \tag{23}$$

Now we can investigate expressions for X_k. Note that the trigonometric functions in the denominator are conveniently cancelled out:

$$X_k \quad = \quad \frac{1}{4} \sum_{i=0}^{31} x_{0,i} \cos \frac{(2i+1)(2k+1)\pi}{128}, \quad k = 0 \ldots 15 \tag{24}$$

$$X_{31-k} \quad = \quad \frac{1}{4} \sum_{i=0}^{31} (-1)^i x_{0,i} \sin \frac{(2i+1)(2k+1)\pi}{128}, \quad k = 0 \ldots 15 \tag{25}$$

In (25), we change the variable: $k = 31 - l$ and obtain for $l = 16 \ldots 31$:

$$\begin{aligned} X_l &= \frac{1}{4} \sum_{i=0}^{31} (-1)^i x_{0,i} \sin \frac{(2i+1)(64-(2l+1))\pi}{128} \\ &= \frac{1}{4} \sum_{i=0}^{31} (-1)^i x_{0,i} \sin \frac{(2i+1) \cdot 64\pi}{128} \cos \frac{(2i+1)(2l+1)}{128} \\ &= \frac{1}{4} \sum_{i=0}^{31} x_{0,i} \cos \frac{(2i+1)(2l+1)}{128} \end{aligned} \tag{26}$$

By comparing (24) with (26), one can conclude that the following expression is valid for the whole range of $k = 0 \ldots 31$:

$$X_k = \frac{1}{4} \sum_{i=0}^{31} x_{0,i} \cos \frac{(2i+1)(2k+1)\pi}{128} \tag{27}$$

Since the code only assigns and shifts the values of `raXin[]`, we can write:

$$X_{k+32j} = \frac{1}{4}\sum_{i=0}^{31} x_{-j,i} \cos\frac{(2i+1)(2k+1)\pi}{128}, \quad k = 0\ldots 31 \tag{28}$$

The code calculates $X_{i+j} - X_{j+k}$ and $-X_{i+j} - X_{j+k}$, where j is divisible by 64 (i.e., $j = 64j'$) and $i + k = 31$. Note to readers: below, k corresponds to `i` in the code, and thus, $31 - k$ in the formulae is `k` in the code. Don't let this confuse you.

$$\begin{aligned} X_{64j'+k} \quad &- \quad X_{64j'+(31-k)} \\ &= \frac{1}{4}\sum_{i=0}^{31} x_{-2j',i}\left(\cos\frac{(2i+1)(2k+1)\pi}{128} - \cos\frac{(2i+1)(63-2k)\pi}{128}\right) \\ &= -\frac{1}{2}\sum_{i=0}^{31} x_{-2j',i}\sin\frac{(2i+1)\pi}{4}\sin\frac{(2i+1)(2k-31)\pi}{128} \end{aligned} \tag{29}$$

Since the last factor will also go into the final answer, in order to discuss its relation to the DCT, we rewrite the above in terms of cosines:

$$\begin{aligned} X_{64j'+k} \quad &- \quad X_{64j'+(31-k)} \\ &= -\frac{1}{2}\sum_{i=0}^{31} x_{-2j',i}\sin\frac{(2i+1)\pi}{4}\sin\frac{(2i+1)(2k+33-64)\pi}{128} \\ &= \frac{1}{2}\sum_{i=0}^{31} x_{-2j',i}(-1)^i\sin\frac{(2i+1)\pi}{4}\cos\frac{(2i+1)(2k+33)\pi}{128} \\ &= \frac{1}{2}\sum_{i=0}^{31} x_{-2j',i}\cos\frac{(2i+1)\pi}{4}\cos\frac{(2i+1)(2k+33)\pi}{128} \end{aligned} \tag{30}$$

Analogously,

$$\begin{aligned} -X_{64j'+k} \quad &- \quad X_{64j'+(31-k)} \\ &= \frac{1}{4}\sum_{i=0}^{31} x_{-2j',i}\left(-\cos\frac{(2i+1)(2k+1)\pi}{128} - \cos\frac{(2i+1)(63-2k)\pi}{128}\right) \\ &= \frac{1}{2}\sum_{i=0}^{31} x_{-2j',i}\cos\frac{(2i+1)\pi}{4}\cos\frac{(2i+1)(2k+97)\pi}{128} \end{aligned} \tag{31}$$

It makes sense to introduce the following quantities instead of the original inputs $x_{-j,i}$:

$$\bar{x}_{-j,i} = x_{-j,i}\cos\frac{(2i+1)\pi}{4} \tag{32}$$

As one can see, this amounts to dividing by $\sqrt{2}$, and, if $i = 1$ or 2 (mod 4), flipping the sign.

Since the code clears Z_k at the end for $k = 32\ldots 63$, we immediately obtain that, just before the clearing,

$$Z_{k+32} = \frac{1}{2}\sum_{j'=0}^{7} C_{64j'+32+k}\sum_{i=0}^{31}\bar{x}_{-2j',i}\cos\frac{(2i+1)(2k+97)\pi}{128}, \quad k = 0\ldots 31 \tag{33}$$

Then these values are shifted in the `raZ[]` buffer, so, before the addition, on the next call we have:

$$Z_k^{(0)} = \frac{1}{2} \sum_{j'=0}^{7} \sum_{i=0}^{31} C_{64j'+32+k} \bar{x}_{-2j'-1,i} \cos \frac{(2i+1)(2k+97)\pi}{128}, \quad k = 0 \ldots 31 \tag{34}$$

Then, the following quantity is added to the buffer:

$$\Delta Z_k = \frac{1}{2} \sum_{j'=0}^{7} \sum_{i=0}^{31} C_{64j'+k} \bar{x}_{-2j',i} \cos \frac{(2i+1)(2k+33)\pi}{128} \tag{35}$$

One can already see that the DCA transform is not a common windowed MDCT. Indeed, Z_k at any given moment of time depends on 16 previous inputs (i.e., has 16 overlapping "windows"), while the usual windowed MDCT has only two overlapping windows covering each point in time.

It is convenient to introduce the "deobfuscated coefficients":

$$\bar{C}_k = \left\{ \begin{array}{ll} C_k, & k = (128j) \ldots (128j+63) \\ -C_k, & k = (128j+64) \ldots (128j+127) \end{array} \right. , \quad j = 0 \ldots 3 \tag{36}$$

One can easily check that Z_k can then be written as the following sum, which is the main result of this paper:

$$Z_k = \frac{1}{2} \sum_{j''=0}^{15} \sum_{i=0}^{31} \bar{x}_{-j'',i} \bar{C}_{32j''+k} \cos \frac{(2i+1)(2(32j''+k)+33)\pi}{128} \tag{37}$$

So, the following 32 kernels (enumerated by i) are important for the transform:

$$\bar{K}_{j,i} = \bar{C}_j \cos \frac{(2i+1)(2j+33)\pi}{128} \tag{38}$$

This structure is similar to the one mentioned in [2] for a pseudo-QMF cosine modulation filter bank. Their definition, adapted to our notation, is:

$$\bar{K}_{j,i} = \bar{C}_j \cos \left(\left(i + \frac{1}{2} \right) \left(j + \frac{1}{2} - \frac{M}{2} \right) \frac{\pi}{N} + \phi_i \right) \tag{39}$$

where $M = 512$ is the prototype filter length, $N = 32$ is the number of bands, and the following condition must hold:

$$\phi_{i+1} - \phi_i = (2r+1)\frac{\pi}{2} \tag{40}$$

By direct comparison of (38) with (39), one finds:

$$\phi_i = \left(i + \frac{1}{2} \right) \frac{17\pi}{2} \tag{41}$$

and

$$\phi_{i+1} - \phi_i = \frac{17\pi}{2} \tag{42}$$

i.e., indeed, an odd multiple of $\pi/2$. Thus, (40) holds. One can also easily verify property (2) in [2] by "deobfuscating" (as defined in (36)) both sets of subband filter coefficients in the DTS specification and calculating the Fourier transform of the result.

Conclusion: the subband transform in the DTS specification is, up to a constant factor, a pseudo-QMF cosine modulation filter bank combined with the sign flip of subband inputs whose number is congruent to 1 or 2 modulo 4.

3 Optimizations

Algorithms exist that calculate IMDCT quickly. It is a good idea to use them for calculating the DCA subband transform. However, to do so, one must group the identical cosine factors.

References

[1] DTS Coherent Acoustics; Core and Extensions. ETSI TS 102 114 V1.2.1 (2002-12) (search for DTS Coherent Acoustics)

[2] Smith, Julius O. Spectral Audio Signal Processing, March 2007 Draft, http://ccrma.stanford.edu/ jos/sasp/, online book, accessed 2008-09-07. Pseudo-QMF Cosine Modulation Filter Bank