Mezzanine Compression for HDTV

R.T. Russell
Abstract

The 1080p50 and 1080p60 standards are ideal for use as studio acquisition formats, however a bit-rate of 3 Gb/s is required to transport them in an uncompressed form. A compression method is described which makes it possible to encode a 1080p50 or 1080p60 signal into a format that is compatible with the 1080i standard and suitable for transport via HD-SDI at 1.5 Gb/s. A novel feature is that the compressed signal is recognisable when displayed as 1080i, for ease of monitoring and identification. When that feature is not required a compression ratio of 2.5:1 is achievable.

This document was originally published in the Proceedings of the International Broadcasting Convention, September 2005.
MEZZANINE COMPRESSION FOR HDTV

R. T. Russell

BBC Research & Development Department, U.K.

ABSTRACT
The 1080p50 and 1080p60 standards are ideal for use as studio acquisition formats, however a bit-rate of 3 Gb/s is required to transport them in an uncompressed form. A compression method is described which makes it possible to encode a 1080p50 or 1080p60 signal into a format that is compatible with the 1080i standard and suitable for transport via HD-SDI at 1.5 Gb/s. A novel feature is that the compressed signal is recognisable when displayed as 1080i, for ease of monitoring and identification. When that feature is not required a compression ratio of 2.5:1 is achievable.

INTRODUCTION
The 1080p50 and 1080p60 standards are ideal for use as studio acquisition formats, because they combine a high spatial resolution (1920 pixels x 1080 lines) with a high temporal resolution (50 or 60 frames per second, progressive scan). In addition they can be easily converted to either of the common transmission formats, 1080i and 720p. However these standards require approximately twice the bit-rate of the transmission formats to carry them in an uncompressed form (3 Gb/s rather than 1.5 Gb/s).

Various proposals have been made for dealing with this problem, including dual-link HD-SDI (multiplexing the 3 Gb/s data between two 1.5 Gb/s circuits), a 3 Gb/s Serial Digital Interface standard and the use of high-speed data over twisted-pair cabling. This paper describes an alternative approach involving mild compression (bit-rate reduction), making it possible to encode a 1080p50/60 signal into a format that is compatible with the 1080i standard and therefore suitable for transport via HD-SDI at 1.5 Gb/s. Such a signal could be carried by existing HD cabling and routing infrastructures in a broadcast centre, and would have the additional convenience of permitting the use of existing embedded audio and ancillary data standards within the signal.

This work is the subject of two patent applications (Ref. 1).

CHARACTERISTICS
The proposed compression algorithm has a number of properties which are desirable for this application:

- Low-delay coding and decoding (total codec delay 8 lines) eliminates picture-sound synchronization problems.
- Low-loss compression provides visually near-perfect reproduction.
- Multi-generation compression adds negligible additional loss.
- Small picture blocks limit the propagation of errors and avoid spatial cross-modulation (e.g. noise in one part of a picture affecting another part).
- Simple algorithm should be relatively easy to implement in hardware.
In addition the proposed algorithm provides a means to make the compressed data recognizable if it is displayed as a standard 1080i signal. This is achieved by encoding a coarsely quantized, interlaced, version of the picture in the most-significant bits of the compressed signal. Although this ‘compatible’ picture is noisy it is good enough to allow identification and monitoring using existing HDTV displays.

**COMPRESSION PROCESS**

**Block Structure**

Each frame is divided into intra-coded macroblocks of 16 (luminance) pixels x 4 TV lines. Each macroblock contains 64 luminance pixels, 32 Cb chrominance pixels and 32 Cr chrominance pixels. For a 1080p frame this results in 120 x 270 = 32400 macroblocks in total. The height of the macroblock determines the minimum delay through the coder or decoder (i.e. four lines).

**Integer Transforms**

Each macroblock is subdivided into 4-pixel x 4-line transform blocks, so in each macroblock there are 4 luminance transform blocks, 2 Cb transform blocks and 2 Cr transform blocks. Each 4x4 block is transformed into frequency space using the following Integer transform \( R = TXT^T \) (Ref. 2):

\[
\begin{bmatrix}
    r_{00} & r_{01} & r_{02} & r_{03} \\
    r_{10} & r_{11} & r_{12} & r_{13} \\
    r_{20} & r_{21} & r_{22} & r_{23} \\
    r_{30} & r_{31} & r_{32} & r_{33}
\end{bmatrix}
= \begin{bmatrix}
    1 & 1 & 1 & 1 \\
    2 & 1 & -1 & -2 \\
    1 & -1 & -1 & 1 \\
    1 & -2 & 2 & -1
\end{bmatrix}
\begin{bmatrix}
    X_{00} & X_{01} & X_{02} & X_{03} \\
    X_{10} & X_{11} & X_{12} & X_{13} \\
    X_{20} & X_{21} & X_{22} & X_{23} \\
    X_{30} & X_{31} & X_{32} & X_{33}
\end{bmatrix}
\begin{bmatrix}
    1 & 2 & 1 & 1 \\
    1 & 1 & -1 & -2 \\
    1 & -1 & -1 & 2 \\
    1 & -2 & 1 & -1
\end{bmatrix}
\]

For the purposes of the computation the 10-bit input pixels (luminance and chrominance) are assumed to be signed values in the range –512 to +511 so ‘zero’ luminance is mid grey; this is the convention used in BBC R&D’s ‘Kingswood format’ picture files. The transform has a maximum gain of 36.

**Scaling**

The 4x4 outputs from the Integer transforms are scaled and rounded according to the following algorithm:

\[
s_{ij} = \text{Sign}(r_{ij}) \times ((\text{Abs}(r_{ij} \times w_{ij}) + 32768) >> 16) \quad \text{..........................(2)}
\]

where:

\[
\begin{bmatrix}
    W_{00} & W_{01} & W_{02} & W_{03} \\
    W_{10} & W_{11} & W_{12} & W_{13} \\
    W_{20} & W_{21} & W_{22} & W_{23} \\
    W_{30} & W_{31} & W_{32} & W_{33}
\end{bmatrix}
= \begin{bmatrix}
    16384 & 10486 & 16384 & 10486 \\
    16384 & 10486 & 16384 & 10486 \\
    16384 & 10486 & 16384 & 10486 \\
    16384 & 10486 & 16384 & 10486
\end{bmatrix}
\begin{bmatrix}
    1 & 2 & 1 & 1 \\
    1 & 1 & -1 & -2 \\
    1 & -1 & -1 & 2 \\
    1 & -2 & 1 & -1
\end{bmatrix}
\]

Note that the multipliers are approximations to the following constants (Ref. 2):

\[
16384 \approx 2^{22}/16/16
\]
\[
10486 \approx 2^{22}/20/20
\]
\[
6711 \approx 2^{22}/25/25
\]
Hadamard Transforms

The DC terms \((s_{00})\) of the scaled Integer transforms, that is four values per macroblock from the luminance and two values each per macroblock from Cb and Cr, are further processed using 1-dimensional Hadamard transforms as follows:

**Luminance:**
\[
\begin{align*}
    h_a &= s_a + s_b + s_c + s_d \\
    h_b &= s_a + s_b - s_c - s_d \\
    h_c &= s_a - s_b + s_c - s_d \\
    h_d &= s_a - s_b - s_c + s_d
\end{align*}
\]

**Chrominance:**
\[
\begin{align*}
    h_a &= s_a + s_b \\
    h_b &= s_a - s_b
\end{align*}
\]

These transforms exploit the probable correlation between the DC terms in adjacent transform blocks, and result in three aggregated DC values per macroblock (Y, Cb, Cr) which are sent, uncompressed, as 10-bit numbers in the compressed data. This results in the DC terms being accurately reproduced at the decoder and reduces block artefacts.

Quantisation

To control the overall bit rate, thus ensuring that the compressed signal fits into the available data capacity, a variable degree of quantising is applied independently to each macroblock. The quantising is applied to all the non-zero transform coefficients in the macroblock, luminance and chrominance, except the three aggregated DC terms \(h_a\) (Y, Cb and Cr).

Since the only purpose is to control the bit rate, it is undesirable to apply any ineffective quantising, i.e. the addition of quantisation noise without any consequent reduction in the total number of coded bits. This is a shortcoming of the quantising schemes used by some other compression techniques.

To avoid this shortcoming the following constraints are imposed:

1. Quantisation consists always of bit shifts, i.e. divisions by powers of two.
2. The variable-length code chosen to encode the quantised coefficients has the property that dividing the coefficient value by two is guaranteed to result in a reduction in the number of coded bits.

The combination of these characteristics means that whenever an additional degree of quantisation is applied (i.e. a coefficient is divided by a greater power of two) the total number of bits is guaranteed to fall. It also helps achieve the property of adding very little or no loss on a second or subsequent generation.

The simplest approach to such quantising would be to divide all (non-zero) coefficients in the macroblock by the same power of two, and signal that number to the decoder. However such a coarse control could result in a much greater degree of quantisation noise than necessary to meet the bit budget.
What is needed is a method of quantising different coefficients to different degrees, without the overhead of signalling additional information to the decoder. The way this is achieved is as follows:

1. The degree of quantisation is controlled by a 5-bit unsigned number $q$, taking a value from 0 to 31.
2. The most-significant 3 bits of $q$, being a value in the range 0 to 7, determine a pair of powers of two by which each coefficient in a macroblock may be divided, as follows:

<table>
<thead>
<tr>
<th>$q$</th>
<th>Divisor</th>
<th>Divisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 or 2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2 or 4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4 or 8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>8 or 16</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16 or 32</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>32 or 64</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>64 or 128</td>
<td>128</td>
</tr>
<tr>
<td>7</td>
<td>128 or 256</td>
<td>256</td>
</tr>
</tbody>
</table>

Table 1 – Coarse divisor selection

3. The least-significant 2 bits of $q$, in conjunction with the value of the coefficient itself, determine which of the two possible divisors is used. A three bit number in the range 4 to 7 is derived from the coefficient value as follows:

(a) Take the absolute value of the coefficient.
(b) If the value is greater than 7 keep dividing by two until 7 or less.
(c) If the value is less than 4 keep multiplying by two until 4 or more.

The following table determines whether the smaller or larger divisor is used, where the rows correspond to the least-significant 2 bits of $q$ and the columns to the number derived from the coefficient:

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>smaller</td>
<td>smaller</td>
<td>smaller</td>
<td>smaller</td>
</tr>
<tr>
<td>1</td>
<td>larger</td>
<td>smaller</td>
<td>smaller</td>
<td>smaller</td>
</tr>
<tr>
<td>2</td>
<td>larger</td>
<td>larger</td>
<td>smaller</td>
<td>smaller</td>
</tr>
<tr>
<td>3</td>
<td>larger</td>
<td>larger</td>
<td>smaller</td>
<td>smaller</td>
</tr>
</tbody>
</table>

Table 2 – Fine divisor selection

So if the least-significant 2 bits of $q$ are zero the smaller of the two divisors is always used, but otherwise which divisor is used depends on the value of the coefficient.

In this way fine control over the degree of quantisation is achieved, such that the equivalent quantisation factor doubles for every increment of 4 in $q$. 
In practice this is complicated by two factors:

1. To minimise the added quantisation noise the coefficient values are rounded rather than simply truncated.

2. The decoder sees only the quantised value, which may not necessarily give rise to the same three-bit number (4-7) as that derived from the original coefficient.

As a result the actual algorithm used is slightly more complex.

The following process is used to determine the appropriate value of $q$ to use for each macroblock:

(a) Initialise $q$ to zero, corresponding to minimum quantising.

(b) Quantise all the (non-DC) coefficients in the macroblock according to the value of $q$, where the most-significant three bits of $q$ determine a pair of possible divisors and the least-significant two bits of $q$ (in association with the coefficient value) determine which of the two divisors to use.

(c) Determine the total number of bits needed to code the quantised macroblock (see below).

(d) If the number of bits exceeds the available capacity (512 bits) increment $q$.

(e) Repeat (b), (c) and (d) until the coded macroblock fits in the available space.

**Variable Length Coding**

The quantised coefficients (except the three DC terms) are coded according to the following signed Exp-Golomb code (Ref. 3):

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
</tr>
<tr>
<td>-1</td>
<td>011</td>
</tr>
<tr>
<td>2</td>
<td>00100</td>
</tr>
<tr>
<td>-2</td>
<td>00101</td>
</tr>
<tr>
<td>3</td>
<td>00110</td>
</tr>
<tr>
<td>-3</td>
<td>00111</td>
</tr>
<tr>
<td>4</td>
<td>000100</td>
</tr>
<tr>
<td>-4</td>
<td>000101</td>
</tr>
<tr>
<td>5</td>
<td>000100</td>
</tr>
<tr>
<td>-5</td>
<td>000101</td>
</tr>
<tr>
<td>6</td>
<td>000110</td>
</tr>
<tr>
<td>-6</td>
<td>000111</td>
</tr>
</tbody>
</table>

Table 3 – Signed Exp-Golomb code
Packet Structure

Each macroblock is encoded into a data packet of 512 bits, corresponding to a 2.5:1 compression compared to the original 10-bits-per-pixel input. The value of $q$ selected for a macroblock is the smallest that results in the size of the compressed data not exceeding 512 bits.

The structure of each packet is as follows:

1. Quantisation code $q$, 5 bits, MSB first.
2. DC luminance coefficient divided by 16 and rounded, 10 bits, MSB first.
3. DC Cb chrominance coefficient divided by 8 and rounded, 10 bits, MSB first.
4. DC Cr chrominance coefficient divided by 8 and rounded, 10 bits, MSB first.
5. Remaining luminance coefficients, signed Exp-Golomb coded, in the order shown below:

<table>
<thead>
<tr>
<th>Y</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>1</th>
<th>5</th>
<th>9</th>
<th>13</th>
<th>2</th>
<th>6</th>
<th>10</th>
<th>14</th>
<th>3</th>
<th>7</th>
<th>11</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>17</td>
<td>21</td>
<td>25</td>
<td>29</td>
<td>18</td>
<td>22</td>
<td>26</td>
<td>30</td>
<td>19</td>
<td>23</td>
<td>27</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>36</td>
<td>40</td>
<td>44</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>45</td>
<td>34</td>
<td>38</td>
<td>42</td>
<td>46</td>
<td>35</td>
<td>39</td>
<td>43</td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td>52</td>
<td>56</td>
<td>60</td>
<td>49</td>
<td>53</td>
<td>57</td>
<td>61</td>
<td>50</td>
<td>54</td>
<td>58</td>
<td>62</td>
<td>51</td>
<td>55</td>
<td>59</td>
<td>63</td>
</tr>
</tbody>
</table>

Table 4 – Luminance coefficient ordering

6. Remaining chrominance coefficients, signed Exp-Golomb coded, in the order shown below:

<table>
<thead>
<tr>
<th>Cb</th>
<th>3</th>
<th>7</th>
<th>11</th>
<th>1</th>
<th>5</th>
<th>9</th>
<th>13</th>
<th>Cr</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>2</th>
<th>6</th>
<th>10</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>19</td>
<td>23</td>
<td>27</td>
<td>17</td>
<td>21</td>
<td>25</td>
<td>29</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>18</td>
<td>22</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>35</td>
<td>39</td>
<td>43</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>45</td>
<td>32</td>
<td>36</td>
<td>40</td>
<td>44</td>
<td>34</td>
<td>38</td>
<td>42</td>
<td>46</td>
</tr>
<tr>
<td>47</td>
<td>51</td>
<td>55</td>
<td>59</td>
<td>49</td>
<td>53</td>
<td>57</td>
<td>61</td>
<td>48</td>
<td>52</td>
<td>56</td>
<td>60</td>
<td>50</td>
<td>54</td>
<td>58</td>
<td>62</td>
</tr>
</tbody>
</table>

Table 5 – Chrominance coefficient ordering

Trailing zero coefficients in the packet are not counted when determining whether the data fits in the available space. Such coefficients need not be stored if the packet is already full (they would otherwise have needed one extra bit of capacity per coefficient). It is only for this reason that the order of the coefficients is significant. In particular where there is no chrominance (all chrominance coefficients are zero) the entire data packet is available for coded luminance.

It may be beneficial to fill any unused data bits with random values, to minimise the visibility of the data on the compatible signal.

When used to compress a 1080p50 or 1080p60 input to a 1080i-compatible output, the data packet is transported in the least-significant 8 bits of the (10-bit) video, requiring 64 consecutive video samples (32 luminance and 32 chrominance). Therefore 60 packets can be contained within a single TV-line of ‘interlaced’ output. The first bit of the data packet is the most-significant bit of the first 8-bit value.
**Compatible Picture**

The two most-significant bits of the 10-bit output can be used to carry a ‘compatible’ interlaced version of the input picture, so that if the compressed signal is viewed as if it were standard 1080i video the content will be recognisable, although noisy. This makes it possible to monitor the signal for the purposes of identification and to give confidence that the coder is working.

The steps to achieve this are as follows:

1. An interlaced version of the input is created, typically by averaging pairs of consecutive progressive lines. The phase of the line pairing must be different on even fields and odd fields in order to achieve the required interlaced structure.

2. Each 10-bit pixel value is modified by subtracting the 8-bit value of the coded data packet corresponding to that pixel's position and adding a two-dimensional halftone dither. After the addition and subtraction the result must be limited to a valid range. The dither consists of a repeating 8x8 pattern of values as follows:

   | 0 | 128 | 32 | 160 | 8 | 136 | 40 | 168 |
---|----|-----|----|-----|---|-----|----|-----|
| 192 | 64 | 224 | 96 | 200 | 72 | 232 | 104 |
| 48 | 176 | 16 | 144 | 56 | 184 | 24 | 152 |
| 240 | 112 | 208 | 80 | 248 | 120 | 216 | 88 |
| 12 | 140 | 44 | 172 | 4 | 132 | 36 | 164 |
| 204 | 76 | 236 | 108 | 196 | 68 | 228 | 100 |
| 60 | 188 | 28 | 156 | 52 | 180 | 20 | 148 |
| 252 | 124 | 220 | 92 | 244 | 116 | 212 | 84 |

   **Table 6 – 2D halftone values**

   The dither is added independently to the luminance, Cb chrominance and Cr chrominance components, where the row selected is determined by the line number evaluated modulo-8, and the column selected is determined by the pixel number (0-1919 for luminance, 0-959 for chrominance) evaluated modulo-8.

3. The least-significant 8-bits of the resulting pixel value are discarded and replaced by the 8-bit value from the coded data packet.

4. If the final 10-bit value is a forbidden TRS code (Ref. 4), the most-significant two bits are modified:

   
   00 000000 xx  is modified to  01 000000 xx
   11 111111 yy  is modified to  10 111111 yy
The effect of the halftone dither can be seen from the following illustrations (magnified so the individual pixels are visible):

Figure 1 - Original linear ramp

Figure 2 - Ramp quantised to two bits

Figure 3 - Ramp quantised to two bits with 2D halftone dither

Figure 4 - Quantised ramp with dither and random data in LSBs

DECOMPRESSION PROCESS

In the interests of brevity details of the decompression process are not given in this paper, but it essentially involves reversing the steps in the compression process (apart from those concerned with generating the compatible interlaced signal and the need to determine the degree of quantisation).
TEST RESULTS

The compression method has been tested using both still pictures and progressively-scanned moving sequences. The moving sequences are more critical because quantising noise and aliasing can often be masked by detail in a still picture. The results have been assessed both subjectively and by means of PSNR measurements. The subjective tests demonstrated that, for the material used, the artefacts were generally invisible even when a difference picture was used to draw attention to them. The PSNR measurements are presented below.

<table>
<thead>
<tr>
<th>Picture</th>
<th>PSNR (luminance)</th>
<th>PSNR(chrominance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newpat PM</td>
<td>55.5 dB</td>
<td>57.6 dB</td>
</tr>
<tr>
<td>Kiel Harbour (1080)</td>
<td>51.4 dB</td>
<td>51.6 dB</td>
</tr>
<tr>
<td>Test Card W</td>
<td>56.9 dB</td>
<td>60.0 dB</td>
</tr>
<tr>
<td>Boy with toys</td>
<td>52.2 dB</td>
<td>51.9 dB</td>
</tr>
<tr>
<td>Dick</td>
<td>51.2 dB</td>
<td>50.8 dB</td>
</tr>
<tr>
<td>Boat</td>
<td>52.7 dB</td>
<td>52.3 dB</td>
</tr>
<tr>
<td>Tree</td>
<td>48.4 dB</td>
<td>47.6 dB</td>
</tr>
<tr>
<td>Tree (10th generation)</td>
<td>48.0 dB</td>
<td>47.5 dB</td>
</tr>
</tbody>
</table>

Table 7 – Still picture PSNR measurements

<table>
<thead>
<tr>
<th>Sequence</th>
<th>PSNR (luminance)</th>
<th>PSNR (chrominance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toys &amp; Calendar (720)</td>
<td>50.5 - 51.1 dB</td>
<td>49.4 - 49.8 dB</td>
</tr>
<tr>
<td>Yosemite (720)</td>
<td>50.1 - 51.1 dB</td>
<td>49.4 - 50.0 dB</td>
</tr>
<tr>
<td>White City (720)</td>
<td>50.4 - 50.5 dB</td>
<td>49.4 - 49.5 dB</td>
</tr>
<tr>
<td>BBCdisk</td>
<td>49.0 - 50.0 dB</td>
<td>48.7 - 49.5 dB</td>
</tr>
<tr>
<td>Football</td>
<td>48.2 - 49.0 dB</td>
<td>47.8 - 48.5 dB</td>
</tr>
<tr>
<td>Skate</td>
<td>52.1 - 53.3 dB</td>
<td>51.7 - 53.0 dB</td>
</tr>
<tr>
<td>Panslow</td>
<td>49.9 – 51.5 dB</td>
<td>50.7 – 52.0 dB</td>
</tr>
<tr>
<td>Panslow (7th gen)</td>
<td>41.5 – 43.2 dB</td>
<td>42.9 – 44.3 dB</td>
</tr>
</tbody>
</table>

Table 8 – Progressive sequence PSNR measurements

Figures 5 to 10 below show some sample pictures, illustrating the appearance of the ‘compatible’ coded picture and the nature of the difference between the original and decoded pictures. A gain of 16 has been applied to the difference pictures.

\[1\] With pixel shifts between each generation.
Figure 5 – Compatible coded picture

Figure 6 – Decoded picture, 1st generation

Figure 7 – Difference picture, 1st generation (gain x16)
Figure 8 – Compatible coded picture

Figure 9 – Decoded picture, after 7 generations with pixel shifts

Figure 10 - Difference picture, after 7 generations (gain x16)
IMPLEMENTATION

It is anticipated that the suggested algorithm could be prototyped within a single inexpensive FPGA (Field Programmable Gate Array) device, certainly in the case of the decoder and probably in the case of the encoder. The low delay implies low storage requirements, which should be achievable with the internal resources of a modern FPGA.

One possible implementation would be as an external ‘dongle’ providing conversion from dual HD-SDI to single HD-SDI (encoder) or from single HD-SDI to dual HD-SDI (decoder).

The main practical consideration when deploying a system of this kind would be the delay, which although only 8 TV lines per codec may be significant in respect of video synchronisation, for example at the inputs of a vision mixer. It may be that some equipment could be made tolerant of delay differences of this sort at its inputs, but failing that it might be necessary to equalise the delays on different sources.

CONCLUSIONS

The compression method described shows promise in being able to achieve a 2:1 reduction in bit rate with artefacts below the threshold of visibility for typical 422 material. Further work will be required to assess the performance across a wider range of material and to determine its suitability for the suggested application.

REFERENCES

1. United Kingdom Patent Applications No. 0507105.5 and 0507106.3 (07 April 2005).
2. BS ISO/IEC 14496-10:2003 Information technology – Coding of audio-visual objects – Part 10: Advanced video coding, section 8.6.1.1

ACKNOWLEDGEMENTS

The author would like to thank the British Broadcasting Corporation for permission to publish this paper.