# FAST CONVERSION OF H.264/AVC INTEGER TRANSFORM COEFFICIENTS INTO DCT COEFFICIENTS

R. Marques[1], V. Silva[1,2], S. Faria[1,3], A. Navarro[1,4], P. Assuncao[1,3]
[1]*Instituto de Telecomunicações,*[2]*Universidade de Coimbra - DEEC, 3030-290 Coimbra, Portugal*
[3]*Instituto Politécnico de Leiria- ESTG , Apt 4163, 2411-901 Leiria, Portugal*
[4]*Universidade de Aveiro – DET, 3810-193 Aveiro, Portugal*

Keywords: Transform conversion, video transcoding.

Abstract: In this paper we propose a fast method to convert H.264/AVC 4x4 Integer Transform (IT) to standard Discrete Cosine Transform (DCT for video transcoding applications. We derive the transcoding matrix for converting, simultaneously, in the transform domain, four IT 4x4 blocks into one $8\times 8$ DCT block of coefficients. By exploiting the symmetry properties of the matrix, we show that the proposed conversion method requires fewer operations than its equivalent in the pixel domain. An integer matrix approximation is also proposed. The experimental results show that a negligible error is introduced, while the computational complexity can be significantly reduced.

## 1 INTRODUCTION

The H.264 is a new video coding standard, recently approved by ITU-T and ISO/IEC as International Standard. When compared to earlier video coding standards like H.263, the H.264 video coding tools can provide enhanced compression efficiency. Experimental results show that about 50% of the bitrate can be saved by using H.264 (Sullivan et al. 2004). Given this coding efficiency, H.264 has been adopted by various international consortiums like the Korean Digital Multimedia Broadcasting (DMB), the European Digital Video Broadcasting (DVB) and the 3rd Generation Partnership Project (3GPP) as the standard video codec, and is expected to be extended to other areas of application, such as, the Blu-ray Disc (BD).

Whenever a new standard is adopted, this always gives rise to interoperability problems with legacy systems. In the case of H.264, interoperability with MEPG-2 systems is of particular importance. In general, this is achieved through video transcoding methods (Chuang et al. 2005). However, there are significant differences between the H.264 and other video coding standards, which difficult the transcoding process, e.g., while the common video codecs use the $8\times 8$ Discrete Cosine Transform (DCT) to reduce spatial correlation, H.264 uses either $4\times 4$ or $8\times 8$ Integer Transforms (IT). The latter is only used in Frext profiles (Sullivan et al. 2004).

This paper addresses the problem of converting H.264/AVC $4\times 4$ IT to standard DCT coefficients for video transcoding applications. We derive the conversion matrix in the transform domain and along with a fast algorithm to reduce the number of operations. Then, we introduce an integer matrix approximation to increase computing performance using fixed-point arithmetic.

The organization of this paper is as follows. In section 2, we describe the proposed transform domain IT-to-DCT conversion. In sections 3 and 4 the fast conversion algorithm and its integer approximation are, respectively, described. The experimental results are presented in section 5 and, finally, in section 6 the main conclusions are reported.

## 2 IT-TO-DCT CONVERSION

The complete (two steps) conversion IT-to-DCT is shown in Figure 1. The input is comprised of four $4\times 4$ IT blocks, $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$. The inverse IT is



Marques R., Silva V., Faria S., Navarro A. and Assuncao P. (2006).
FAST CONVERSION OF H.264/AVC INTEGER TRANSFORM COEFFICIENTS INTO DCT COEFFICIENTS.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 5-8
DOI: 10.5220/0001572500050008

applied to each block in order to obtain the pixel domain blocks, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. Then, the four pixel domain blocks are combined to form a single $8 \times 8$ block **x** to which the DCT is applied, such that an $8 \times 8$ block of transform coefficients **Y** is obtained. However, a full transform domain conversion (one step approach) is more efficient because complete decoding up to the pixel domain is not required.

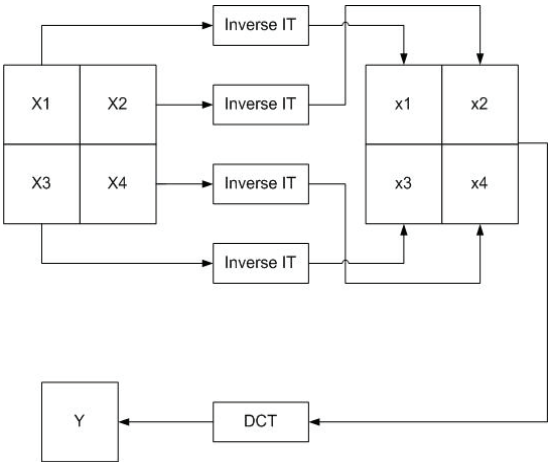


Figure 1: Pixel domain IT-to-DCT conversion.

The proposed transform domain IT-to-DCT conversion is based on simple algebraic matrix relationships (Xin et al. 2004). It is directly applied to an $8 \times 8$ block **X** comprised of four $4 \times 4$ IT blocks, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$, to produce the corresponding $8 \times 8$ DCT block, **Y**. The conversion is given by the following operation,

$$\mathbf{Y} = \mathbf{S} \times \mathbf{X} \times \mathbf{S}^T, \tag{1}$$

where **S** is the transcoding matrix. In order to derive **S**, we have to consider the inverse IT of blocks, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$, given by

$$\mathbf{x}_i = \mathbf{J}\mathbf{X}_i\mathbf{J}^T, \quad 1 \le i \le 4, \tag{2}$$

where **J** is the following matrix (Malvar et al. 2003),

$$\mathbf{J} = \begin{pmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{pmatrix}. \tag{3}$$

If we consider the following matrix, $\mathbf{K} = \begin{pmatrix} \mathbf{J} & 0 \\ 0 & \mathbf{J} \end{pmatrix}$,

then, we can compute **x** in a single step as given by,

$$\mathbf{x} = \mathbf{K} \times \mathbf{X} \times \mathbf{K}^T = \begin{pmatrix} \mathbf{J}\mathbf{X}_1\mathbf{J}^T & \mathbf{J}\mathbf{X}_2\mathbf{J}^T \\ \mathbf{J}\mathbf{X}_3\mathbf{J}^T & \mathbf{J}\mathbf{X}_4\mathbf{J}^T \end{pmatrix}. \tag{4}$$

Since the DCT of an $8 \times 8$ block can be defined as

$$\mathbf{Y} = \mathbf{T} \times \mathbf{x} \times \mathbf{T}^T, \tag{5}$$

where **T** is the DCT kernel matrix, then, it follows that,

$$\mathbf{Y} = \mathbf{T} \times \mathbf{K} \times \mathbf{X} \times \mathbf{K}^T \times \mathbf{T}^T. \tag{6}$$

From (6) we can define the transcoding matrix **S** as,

$$\mathbf{S} = \mathbf{T} \times \mathbf{K}. \tag{7}$$

The structure of matrix **S** is given by,

$$\mathbf{S} = \left(\begin{array}{cccc|cccc} a & 0 & 0 & 0 & a & 0 & 0 & 0 \\ b & c & d & e & -b & c & -d & e \\ 0 & f & 0 & g & 0 & -f & 0 & -g \\ h & i & j & k & -h & i & j & k \\ 0 & 0 & a & 0 & 0 & 0 & a & 0 \\ l & m & n & o & -l & m & -n & o \\ 0 & -g & 0 & f & 0 & g & 0 & -f \\ p & q & r & s & -p & q & -r & s \end{array}\right) \tag{8}$$

with

$$\begin{array}{llll} a = 1.4142 & b = 1.2815 & c = 0.4618 & d = -0.1065 \\ e = 0.0585 & f = 1.1152 & g = 0.0793 & h = -0.45 \\ i = 0.8399 & j = 0.7259 & k = -0.0461 & l = 0.3007 \\ m = -0.4319 & n = 1.0864 & o = 0.5190 & p = -0.2549 \\ q = 0.2412 & r = -0.5308 & s = 0.9875 & \end{array}$$

The shown **S** matrix values are rounded to four decimal places.

# 3 FAST ALGORITHM

The proposed fast IT-to-DCT conversion algorithm is based on the symmetry properties of the **S** matrix shown in (8). As it shall be explained, this characteristic of the **S** matrix is exploited for achieving fast computation of the transform conversion.

Since the conversion defined by (1) is separable, it can be computed by columns followed by rows. If we define *z* as an input 8 point column vector and **Z** its 1D conversion, then, by using the horizontal symmetry of the **S** matrix, we can use the following fast algorithm to compute **Z** as,

$$\begin{aligned} m_1 &= z[1]+z[5] \\ m_2 &= z[1]-z[5] \\ m_3 &= z[2]+z[6] \\ m_4 &= z[3]-z[7] \\ m_5 &= z[4]+z[8] \\ m_6 &= z[4]-z[8] \\ m_7 &= z[3]+z[7] \\ m_8 &= z[2]-z[6] \end{aligned}, \quad (9)$$

$$\begin{aligned} Z[1] &= a\times m_1 \\ Z[2] &= b\times m_2 + c\times m_3 + d\times m_4 + e\times m_5 \\ Z[3] &= f\times m_8 + g\times m_6 \\ Z[4] &= h\times m_2 + i\times m_3 + j\times m_4 + k\times m_5 \\ Z[5] &= a\times m_7 \\ Z[6] &= l\times m_2 + m\times m_3 + n\times m_4 + o\times m_5 \\ Z[7] &= g\times m_8 + f\times m_6 \\ Z[8] &= p\times m_2 + q\times m_3 + r\times m_4 + s\times m_5 \end{aligned}. \quad (10)$$

This algorithm needs 22 multiplications and 22 additions, i.e., a total of 44 operations to perform one 1D conversion. The full 2D fast conversion algorithm needs $8\times 44\times 2 = 704$ operations. Instead, the pixel domain approach needs four inverse IT (320 operations) and one direct DCT (672 operations) which yields a total of 992 operations. (Xin et al. 2004, Lee et al 2005). Thus, the proposed fast algorithm significantly reduces the number of operations (29%) when compared to the pixel domain conversion.

# 4 INTEGER APPROXIMATION

In order to achieve higher computing performance, we have further introduced an integer approximation of the matrix $\mathbf{S}$. This is of particular relevance for fixed-point arithmetic hardware, which is much faster than floating point. The ultimate generation of DSPs operate with clock frequencies of 300MHz for floating-point architectures, while that of fixed-point architecture is about 1GHz (Texas Instruments, 2004).

In order to work with integer arithmetic, we scale the $\mathbf{S}$ matrix by multiplying it by an integer that is a power of 2. To represent each H.264 residual pixel value, we need 9 bits and to perform the IT, we need 11 bits to represent the coefficients. The maximum gain of the 2D $\mathbf{S}$-transcoding matrix is $4.67^2$, which implies that more 5 bits are needed to represent the result of the conversion. Therefore, the scaling factor must be smaller or equal than the square root of $\left(2^{32}-2^{16}\right)=256$. The integer S matrix version is given by, $\mathbf{S}_{\text{int}} = round(256\times\mathbf{S})$, yielding

$$\mathbf{S}_{\text{int}} = \begin{pmatrix} a' & 0 & 0 & 0 & a' & 0 & 0 & 0 \\ b' & c' & d' & e' & -b' & c' & -d' & e' \\ 0 & f' & 0 & g' & 0 & -f' & 0 & -g' \\ h' & i' & j' & k' & -h' & i' & -j' & k' \\ 0 & 0 & a' & 0 & 0 & 0 & a' & 0 \\ l' & m' & n' & o' & -l' & m' & -n' & o' \\ 0 & -g' & 0 & f' & 0 & g' & 0 & -f' \\ p' & q' & r' & s' & -p' & q' & -r' & s' \end{pmatrix}$$

The corresponding $\mathbf{S}_{\text{int}}$ values are given by,

$$\begin{aligned} &a'=362 \quad b'=328 \quad c'=118 \quad d'=-27 \\ &e'=14 \quad f'=285 \quad g'=20 \quad h'=-115 \\ &i'=227 \quad j'=185 \quad k'=-11 \quad l'=75 \\ &m'=-110 \quad n'=278 \quad o'=132 \quad p'=-65 \\ &q'=61 \quad r'=-136 \quad s'=352 \end{aligned}.$$

Since the $\mathbf{S}_{\text{int}}$ matrix symmetries are similar to $\mathbf{S}$, thus, we can also apply the fast algorithm described in section 3.

## 4.1 Multiplierless Implementation

In order to reduce, even more, the computational complexity of the proposed integer conversion algorithm, we may not use hardware multipliers. It is possible to identify in (10) the following multiple constants multiplication boxes,

$$b_1 = m_2\times\begin{bmatrix} b \\ h \\ l \\ p \end{bmatrix}, b_2 = m_3\times\begin{bmatrix} c \\ i \\ m \\ q \end{bmatrix}, b_3 = m_4\times\begin{bmatrix} d \\ j \\ n \\ r \end{bmatrix}$$

$$b_4 = m_5\times\begin{bmatrix} e \\ k \\ o \\ s \end{bmatrix}, b_5 = m_6\times\begin{bmatrix} f \\ g \end{bmatrix}, b_6 = m_8\times\begin{bmatrix} g \\ f \end{bmatrix}$$

which are easily implemented using only elementary operations, i.e., additions, subtractions and shifts

(Puschel et al. 2004), The number of low complexity operations required to compute each multiplier box is shown in Table 1.

Table 1: Number of operations per multiplier block.

| Block | Add/Sub | Shift | Neg |
|---|---|---|---|
| $b_1$ | 5 | 7 | 2 |
| $b_2$ | 5 | 7 | 1 |
| $b_3$ | 6 | 7 | 1 |
| $b_4$ | 4 | 7 | 1 |
| $b_5$ | 3 | 4 | 1 |

Table 2 shows the number of clock cycles required by a general purpose processor (Intel, 2001) to compute each *multiplier block*, (column **Mb**) as well as the conventional multiplier method (column **Mu**). As it can be seen, the number of clock cycles required by the integer fast approximation based on *multiplier blocks* is about 61% of those required by the conventional method.

Table 2: Number of clock cycles per block operations.

| Block | Add/Sub | Shift | Neg | **Mb** | **Mu** |
|---|---|---|---|---|---|
| $b_1$ | 5 | 35 | 2 | **42** | **68** |
| $b_2$ | 5 | 35 | 1 | **41** | **68** |
| $b_3$ | 6 | 35 | 1 | **42** | **68** |
| $b_4$ | 4 | 35 | 1 | **40** | **68** |
| $b_5$ | 3 | 20 | 1 | **24** | **34** |

# 5 EXPERIMENTAL RESULTS

We have evaluated the error introduced by integer approximation of the **S** matrix by comparing both methods described in previous sections.

A set of 3 different grey level images was used (256x256, 8 bit/pel). For each one, the whole image was transformed into $4 \times 4$ IT coefficient blocks. Then, each group of four adjacent $4 \times 4$ IT coefficient blocks are DCT converted by means of two different methods: i) the full precision algorithm described in section 2; ii) the integer approximation described in section 4. The mean squared error (MSE), between both resulting images (pixel domain), was used for evaluating the error introduced by the integer approximation method.

The results are shown in Table 3, where it can be seen that the error due to the integer approximation in the conversion process is actually very small. In fact, the resulting MSE is negligible in practical terms, which proves the usefulness of the proposed method for fast transcoding implementations.

Table 3: MSE of the integer approximation.

| **Image** | Einstein | Smandril | Cameraman |
|---|---|---|---|
| **MSE** | 0.337 | 0.339 | 0.340 |

# 6 CONCLUSIONS

In this paper, we proposed a transform domain approach for fast conversion H.264/AVC 4x4 Integer Transform to standard DCT. We derived the conversion matrix and an efficient algorithm for computing the transform, as well as, a low complexity integer approximation method. The presented results show that the proposed methods are much faster than the pixel domain approach. These methods are suitable for video transcoding applications where fast processing is required.

# REFERENCES

Chuang, S.,Vetro, A., 2005. Video Adaptation: Concepts, Technologies, and Open Issues In *Proceedings of the IEEE*, vol. 93, no. 1, pp 148-158.

Intel, 2001. Intel Pentium 4 Processor Optimization Reference Manual, Order Number 248966

Lee, J., Chung, K., 2005. Quantization/DCT conversion Scheme for DCT-Domain MPEG-2 to H.264/AVC Trascoding. In *IECIE Trans. Commun*., vol E88-B.

Malvar, H., Hallapuro, A., Korczewicz, M., Kerofsky, L.,2003. Low-Complexity Transform and Quantization in H.264/AVC. In *IEEE Transactions on Circuits and Systems for Video Technology*, vol 13, pp 598-603.

Pushel, M, Voronenko, Y., 2004. Multiplierless Constant Multiplication, Spiral Project, Carnegie Mellon University ( http://www.spiral.net).

Sullivan, G., Topiwala, P., Lu, A., 2004. The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. In *SPIE Conference on Applications of Digital Image Processing XXVII*.

Texas Instruments, 2004. TMS 320C600 CPU and Instruction Set Reference Guide. Literature Number SPRU189F.

Xin, J., Vetro, A., Sun, H., 2004. Converting DCT Coefficients to H.264/AVC Transform Coefficients. In *Technical Report of Mitsubishi Electric Lab*. TR-2004-058.