ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementation of Reusable DCT Architectures with Reversible Adders

A Mehjabeen
M.Tech (VLSI & Embedded Systems)
Department of ECE
G Pulla Reddy Engineering College (Autonomous)
Kurnool, A.P, India.

Sri T Thammi Reddy, M.Tech., (Ph.D)
Associate Professor,
Department of ECE
G Pulla Reddy Engineering College (Autonomous)
Kurnool, A.P, India.

Abstract:
In this paper, power efficient architectures for implementation of integer discrete cosine transform (DCT) of the different lengths that can be used in High Efficiency Video Coding (HEVC) have been presented. Efficient constant matrix-multiplication schemes have been presented to derive parallel architectures for a 1-D integer DCT of variable lengths. It also describes about how the proposed structure supports for a reusable structure of DCT for 4, 8, 16, and 32 variable lengths with a throughput of a 32 DCT coefficients per cycle without related to size of a transform. The Reversible logic gates have been used to design the adders which are used in reusable architecture to further reduce the power consumption. From the synthesis and simulation results we have observed that the improvement in power. The proposed architecture will also support in an applications like ultrahigh definition 7680×4320 at 60 frames/s video which is the one of the applications of HEVC.

Key words: ***Discrete cosine transform (DCT), H.265, High Efficiency Video Coding (HEVC), integer discrete cosine trans-form (DCT), video coding.***

I. INTRODUCTION

Discrete Cosine Transforms of different lengths such as 4, 8, 16 and 32 supports High Efficient Video Coding. Therefore flexible hardware architecture is required for DCT of any length for computation. The existing designs for a conventional DCT using constant matrix multiplication CMM and Multiple Constant Multiplications MCM can provide an optimal solution for computation of any of these lengths; the main drawback is they cannot be used as reusable architecture for different lengths of DCT. This drawback can be overcome by constructing a new hardware implementation algorithm for supporting the reusability and lee hardware complexity in computation. We have designed a scalable and reusable architecture for 1-D and a 2-D integer DCTs which can be used for any of these prescribed lengths such as 4,8,16 and 32. The proposed architecture consumes less power.

LITERATURE SURVEY

The efficient integer DCT architectures are proposed for High Efficient Video Coding. HEVC has a one main feature is that it supports a DCT of different sizes like 4, 8, 16 and 32 [1]. DCT is very important in video compression due to its near optimal de-correlation efficiency [2]. In last two decades several variations proposed for integer DCT to reduce the complexity in computation [3]. The new High Efficiency Video Coding (HEVC) H.265 standard [8] was finalized recently to replace AVC/H.264 [9]. In real time implementation they have also proposed some hardware architectures for the integer DCT for HEVC. Author Ahmed *et al* [10] modified the DCT matrix into sparse sub matrices by using lifting scheme to reduce the multiplication. Author Shen *et al* [11] used the multiple constant multiplications (MCM) approach where multiplier is replaced by shift add logic for 4-point and 8-point DCT, and also used normal multipliers with sharing techniques for 16 and 32-point DCTs. Author Park et al [12] have used Chen's factorization of a DCT where the butterfly

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

operation have been implemented by using only multiplexors, adders and shifters. Budagavi and Sze [13] proposed forward and inverse transform after matrix decomposition.

DCT of different lengths such as 4,8,16 & 32 can be used for HEVC. Hence, flexible hardware architecture must be needed DCT of any length computation. The existing designs for a conventional DCT using CMM and MCM can provide an optimal solution for computation of any of these lengths, since they cannot be used has a reusable architecture for different lengths of DCT. By taking this issue into consideration, we proposed a new hardware implementation algorithm for supporting the reusability and lee hardware in computational complexity. We have designed a scalable and reusable architecture for 1-D and a 2-D integer DCTs for an HEVC which can be reused for any of the selected lengths results in same throughput.

The next section describes the algorithms for a hardware implementation of the HEVC integer DCTs of various lengths 4, 8, 16, and 32. Section III describes the design of architecture for the implementation of 4 and 8 point integer DCT with a generalized design of integer DCT of length N, where it can be used for the 16-point and 32-point DCT. Section IV describes the proposed power-efficient designs of transposition buffers for a full-parallel and folded implementations of 2-D Integer DCT. Section V describes the proposed Reversible logic adder using reversible logic gates. Sections VI describes the observed simulation results & compare the synthesis result of proposed architecture with existing architectures.

INTEGER DCT ARCHITECTURES

II Algorithm for Hardware Implementation of Integer DCT

The Joint Collaborative Team-Video Coding (JCT-VC) manages the standardization of HEVC; Core Experiment 10 (CE10) studied the design of core transforms over many meeting cycles [14]. Generally HEVC transform design [15] contains coefficients of size 8-bit [14] but does not support like other computing proposals for full factorization. However it allows for both matrix multiplication and a partial butterfly implementation. This section describes the computation of integer DCT using the partial-butterfly algorithm of [15].

A. Key Features of Integer DCT

An *N*-point integer DCT for an HEVC given in [15] can be computed by partial butterfly approach using an (*N*/2)-point DCT and matrix–vector product of (*N*/2) × (*N*/2) matrix with a (*N*/2)-point vector as

$$\begin{bmatrix} y(0) \\ y(2) \\ \cdot \\ \cdot \\ \cdot \\ y(N-4) \\ y(N-2) \end{bmatrix} = C_{N/2} \begin{bmatrix} a(0) \\ a(2) \\ \cdot \\ \cdot \\ \cdot \\ a(N/2-2) \\ a(N/2-1) \end{bmatrix} \quad (1a)$$

$$\begin{bmatrix} y(1) \\ y(3) \\ \cdot \\ \cdot \\ \cdot \\ y(N-3) \\ y(N-1) \end{bmatrix} = M_{N/2} \begin{bmatrix} b(0) \\ b(1) \\ \cdot \\ \cdot \\ \cdot \\ b(N/2-2) \\ b(N/2-1) \end{bmatrix} \quad (1b)$$

Where

$a(i)=x(i)+x(N-i-1);\ b(i)=x(i)-x(N-i-1)$ (2)

for i = 0, 1,…, N/2−1. X = [*x(0), x(1),…, x(N −1)*] is the input vector and Y = [*y(0), y(1), ···, y(N−1)]* is N-point DCT of X. And $C_{N/2}$ is (N/2)-point integer DCT kernel matrix of a size (N/2) × (N/2). Whereas MN/2 is also a matrix of a size (N/2) × (N/2) and its (i, j)th entry is defined as

$$m_{N/2}^{i,j} = c_N^{2i+1,j} \text{ For } 0 \le i, j \le N/2\text{-}1 \quad (3)$$

Where $c_N^{2i+1,j}$ is the $(2i + 1, j)$th entry of a matrix C_N. Hence note that (1a) could be similarly decomposed and recursively by further using $C_{N/4}$ $M_{N/4}$. Hence referred to the direct implementation of a DCT based on (1)–(3) as the reference algorithm [1].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

B. Hardware Oriented Algorithm

The direct implementation of (1) requires $N^2/4$ + $MUL_{N/2}$ multiplications, and $N^2/4$ + N/2 + $ADD_{N/2}$ additions and 2 shifts where $MUL_{N/2}$ and $ADD_{N/2}$ of (N/2)-point DCT. The computation of (1) may be represented as a CMM problem [16]–[18]. Thus the absolute values of the coefficients in all the rows and a column of matrix M in (1b) are similar, CMM problem can be implemented as a set of N/2 MCMs which result in a highly regular architecture and have implemented for low-complexity architecture. The kernel matrices for four-point, eight-point, 16-point, and 32-point integer DCT for HEVC are given in [15] and 4- and eight point integer DCT are represented respectively as

$$C_4 = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \quad (4)$$

And

$$C_8 = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix} \quad (5)$$

III Architectures for Integer DCT Computation

This section presents the architecture for a computation of integer 4-point DCT. Generalized architecture for integer DCT of higher lengths is also presented in this section

A. Architecture for Four-Point Integer DCT

The architecture for 4-point integer DCT is shown in Fig. 1(a). Which consists of input adder unit (IAU), shift-add unit (SAU) and output adder unit (OAU). An IAU computes the values of a(0), a(1), b(0) and b(1) according to the STAGE-1 of algorithm [1]. The computations of $t_{i,36}$ and $t_{i,83}$ are performed by two SAUs according to the STAGE-2 of the algorithm. Hence computation of $t_{0,64}$ and $t_{1,64}$ will not requires any logic due to shift operation have been replaced in the architecture. The structure of Shift Add Unit is shown in Fig. 1(b) and outputs of the SAU are finally added by the Output Add Unit according to the STAGE-3.

B. Architecture for Integer DCT of Length 8 and Higher Length DCTs

The *N*-point integer DCT architecture based on the algorithm is shown in Figure 2. The architecture consists of four units such as IAU, an (*N*/2)-point integer DCT unit, an SAU and an OAU elements. The IAU computes the values of $a(i)$ and $b(i)$ for i = 0, 1, ...,$N/2 - 1$ according to STAGE-1 of an algorithm of Section II. An SAU provides the results of multiplication for the input samples with DCT coefficient by STAGE-2. Finally, Output Add Unit generates output of DCT from the architecture binary adder tree. Figure 3(a)–(c) represents the structures of IAU, an SAU and a OAU in the case of 8-point integer DCT. To compute the $t_{i,89}$, $t_{i,75}$, $t_{i,50}$, and $t_{i,18}$ for i = 0, 1, 2, and 3 requires Four SAUs according to STAGE-2. Finally the outputs of SAUs are added by two-stage adder tree according to STAGE- 3. The structures of 16- and 32-point integer DCT can also be obtained similarly.

C. Reusable Integer DCT Architectures

The proposed structure shown in Fig.4 can compute one 32-point DCT, two 16-point DCTs, four eight point DCTs, and eight four-point DCTs, even then throughput remains the same as 32 DCT co efficient per cycle irrespective of the desired transform size.

From the figure 5 (a) we observed that there are two (N/2)-point DCT units in the structure of hardware. The input to a one (N/2)-point DCT unit is given as feedback through (N/2) 2:1 MUXes that selects either [a(0), ..., a(N/2 − 1)] or [x(0), ..., x(N/2 − 1)], which depends on whether it is used for N-point DCT computation or for a DCT of a lower size. The other N/2 point DCT takes the input [x(N/2)….x(N-1)] when it is used for computation of DCT of N/2 point or a lower size, or else input reset by array of (N/2) AND gates to disable (N/2) DCT unit. The output of (N/2)

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

point is multiplexed with OAU, which proceeded by SAU and IAU. The N AND gates before IAU are used to disable IAU, SAU & OAU when architecture computes (N/2)-point DCT or lower size. The input control unit, m_N is used to decide the size of DCT computation. For N=32, m_{32} is 2-bits signal that is set to {00},{01},{1,0} and {11} to compute 4,8,16 & 32-point DCT. The control unit generates sel_1 and sel_2. Here sel_1 is used as control signals of N MUXes and input of N AND gates before IAU and sel_2 in Fig 5(a) is used as the input $m_{N/2}$ to two lower size reusable integer DCT units in a recursive manner. The combinational logics to control units are shown in Fig. 5(b) and (c) for N = 16 and 32, respectively. For N = 8, m_8 is 1-bitsignal that is used as sel_1 while sel_2 is no required since four point DCT is the smallest DCT.

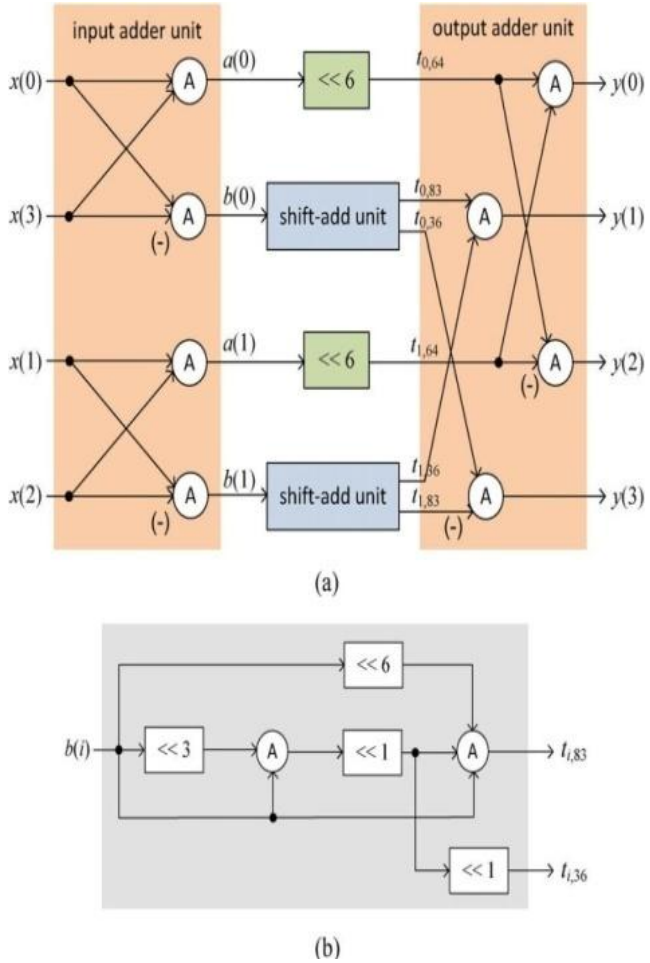

Fig.1. Architecture of four-point integer DCT (a)Four-point DCT architecture (b) Structure of SAU

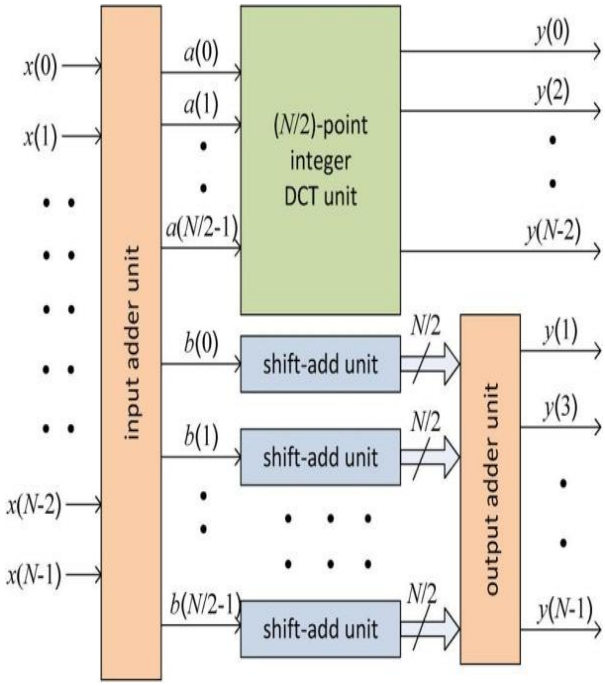

Fig. 2. Generalized architecture for integer DCT of lengths N= 8, 16, and 32.

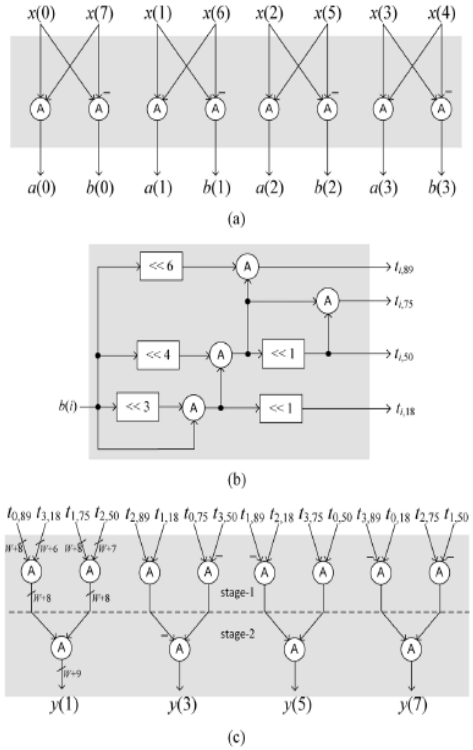

Fig.3. Architecture of eight-point integer DCT (a) Structure of IAU (b) Structure of SAU (c) Structure of OAU

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Fig.4. Reusable Architecture of Integer DCT for N=8, 16 & 32

IV. Structure for 2 -D Integer DCT

The separable property, of an (N × N)-point 2-D integer DCT will be computed by the row-column decomposition technique in two different stages.

1) STAGE-1: N-point 1-D integer DCT is computed for each column of the input matrix with size (N × N) which generates an intermediate output matrix of size (N × N).

2) STAGE-2: N-point 1-D DCT is computed as each row of the intermediate output matrix of size (N × N) which generate required 2-D DCT of size (N × N).

A folded architecture and full-parallel architecture for the 2-D integer DCT, along with the necessary transposition buffer to match them without internal data movement are used.

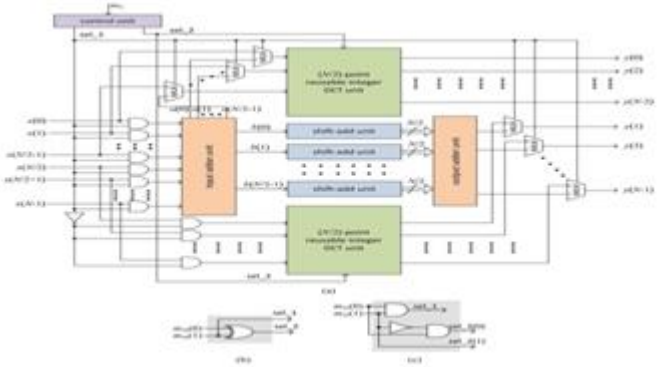

Fig.5. Proposed reusable architecture of integer DCT. (a) Proposed reusable architecture for N = 8, 16, and 32. (b) Control unit for N=16. (c) Control unit for N = 32.

REVERSIBLE ADDERS

V. FULL ADDER USING TWO PERES GATES

Peres Gate

A 3x3 Peres gate is shown in Fig:6. In this the input vector is I (A, B, C) with output vector O (P, Q, R) for which the output is computed as P = A, Q = A xor B and R= AB xor C.

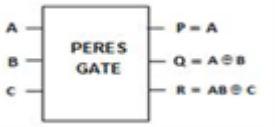

Fig 6: Peres Gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Table1: Truth table of Peres gate

A full adder victimization 2 Peres gates is as shown in Fig 7 .The quantum realization of this shows that its quantum price is eight 2 Peres gates are used. One 4*4 reversible gate referred to as PFAG that is Peres Full Adder Gate with quantum price of eight is employed to appreciate the multiplier factor.

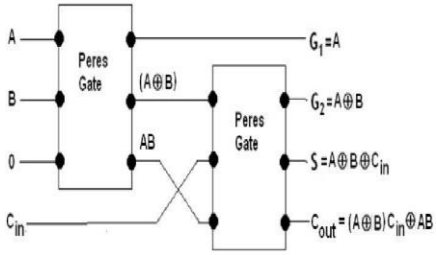

Fig 7: Full adder using Peres Gates

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

VI. SIMULATION AND SYNTHESIS RESULTS

Xilinx ISE Design Suite 12.2 platform is used for the simulation and synthesis of the proposed architecture. The hardware description language Verilog HDL is used for implementation of the proposed architecture.

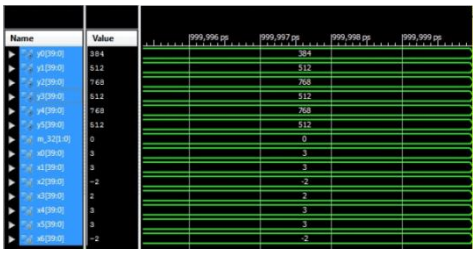

Fig 8: Simulation results of 32 bit DCT output results

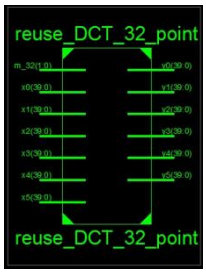

Fig 9: RTL schematic of 32 Point DCT architecture

PARAMETER	32 POINT DCT(EXISTING)	32 POINT DCT (PROPOSED)
No of Slice LUTs	14962	15300
DELAY	44.556 Nano Seconds	45.60 Nano Seconds
Total REAL Time XST completion	217 Seconds	291 Seconds
Total CPU Time XST completion	216.92 Seconds	291.32 Seconds
STATIC POWER	0.118 Watts	0.113 Watts
DYNAMIC POWER	0.148 Watts	0.015 Watts
TOTAL POWER	0.267 Watts	0.128 Watts

Table2: Comparison table of 32 bit existing DCT & proposed DCT architectures

VII. Summary and Conclusion

In this paper, we have proposed power-efficient architectures for the implementation of integer DCT of different lengths that can be used in HEVC. The computation of *N*-point 1-D DCT include (*N*/2)-point 1-D DCT and a vector-matrix multiplication of size (*N*/2) × (*N*/2) with constant matrix. We have shown that the MCM-based architecture is highly regular and involves significantly less energy consumption than the direct implementation of matrix multiplication for odd DCT coefficients. The proposed architecture we used to derive a reusable architecture for DCT which computes the DCT of lengths 4, 8, 16, and 32 with throughput of 32 output coefficients per cycle. We also observed that the power consumption has been reduced from 0.267 Watts to 0.128 Watts by replacing the adders with Reversible adder by reversible logic gates.

Future scope

In future we can extend the length of the DCT to 64 bit and the multiplier used can be replaced by efficient multiplier for further reducing the hardware.

References

[1]Pramod Kumar Meher; Sang Yoon Park; Basant Kumar Mohanty; Khoon Seong Lim; Chuohao Yeo,"Efficient Integer DCT Architectures for HEVC", IEEE Transactions on Circuits and Systems for Video Technology, Volume: 24, 2014, Issue: 1, pp: 168 – 178

[2] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," IEEE Trans. Comput., vol. 100, no. 1, pp. 90–93, Jan. 1974.

[3] W. Cham and Y. Chan, "An order-16 integer cosine transform," IEEETrans. Signal Process., vol. 39, no. 5, pp. 1205–1208, May 1991.

[4] Y. Chen, S. Oraintara, and T. Nguyen, "Video compression using integer DCT," in Proc. IEEE Int. Conf. Image Process., Sep. 2000, pp. 844–845.

[5] J. Wu and Y. Li, "A new type of integer DCT transform radix and its rapid algorithm," in Proc. Int. Conf. Electric Inform. Control Eng., Apr. 2011, pp. 1063–1066.

[6] A. M. Ahmed and D. D. Day, "Comparison between the cosine and Hartley based naturalness

preserving transforms for image watermarking and data hiding," in Proc. First Canad. Conf. Comput. Robot Vision, May
2004, pp. 247–251.

[7] M. N. Haggag, M. El-Sharkawy, and G. Fahmy, "Efficient fast multiplication-free integer transformation for the 2-D DCT H.265 standard," in Proc. Int. Conf. Image Process., Sep. 2010, pp. 3769–3772.

[8] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, High Efficiency Video Coding (HEVC) Text Specification Draft 10 (for FDIS and Consent), JCT-VC L1003, Geneva, Switzerland, Jan. 2013.

[9] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560–576, Jul. 2003.

[10] A. Ahmed, M. U. Shahid, and A. Rehman, "N Point DCT VLSI Architecture for Emerging HEVC Standard," in Proc. VLSI Design, vol. 2012, Article 752024, pp. 1–13, 2012.

[11] S. Shen, W. Shen, Y. Fan, and X. Zeng, "A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards," in Proc. IEEE Int. Conf. Multimedia Expo, Jul. 2012, pp. 788–793.

[12] J.-S. Park, W.-J. Nam, S.-M. Han, and S. Lee, "2-D large inverse transform (16x16, 32x32) for HEVC (high efficiency video coding)," J. Semicond. Technol. Sci., vol. 12, no. 2, pp. 203–211, Jun. 2012. 178 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 24, NO. 1, JANUARY 2014

[13] M. Budagavi and V. Sze, "Unified forward+inverse transform architecture for HEVC," in Proc. Int. Conf. Image Process., Sep. 2012, pp. 209–212.

[14] P. Topiwala, M. Budagavi, A. Fuldseth, R. Joshi, and E. Alshina. (2011, Nov.). JCTVC-G040, CE10: Summary Report on Core Transform Design [Online]. Available: http://phenix.int-evry.fr/jct/doc end user/ documents/7 Geneva/wg11/JCTVC-G040-v3.zip

[15] A. Fuldseth, G. Bjøntegaard, M. Budagavi, and V. Sze. (2011, Nov.). JCTVC-G495, CE10: Core Transform Design for HEVC: Proposal for Current HEVC Transform [Online]. Available: http://phenix.int-evry.fr/ jct/doc end user/documents/7 Geneva/wg11/JCTV%C-G495-v2.zip

[16] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 15, no. 2, pp. 151–165, Feb. 1996.

[17] M. D. Macleod and A. G. Dempster, "Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers," Electron. Lett., vol. 40, no. 11, pp. 651–652, May 2004.

[18] N. Boullis and A. Tisserand, "Some optimizations of hardware multiplication by constant matrices," IEEE Trans. Comput., vol. 54, no. 10, pp. 1271–1282, Oct. 2005.