Hindawi Publishing Corporation
VLSI Design
Volume 2012, Article ID 209208, 12 pages
doi:10.1155/2012/209208

Research Article

Optimized Architecture Using a Novel Subexpression Elimination on Loeffler Algorithm for DCT-Based Image Compression

Maher Jridi,[1] Ayman Alfalou,[2] and Pramod Kumar Meher[3]

[1] *Vision Department, L@bIsen, ISEN–Brest, CS 42807, 29228 Brest Cedex 2, France*
[2] *Vision Department, L@bIsen, ISEN–Brest, CS 42807, 29228 Brest Cedex2, France*
[3] *Department of Embedded Systems, Institute for Infocomm Research, Singapore 138632*

Correspondence should be addressed to Maher Jridi, maher.jridi@isen.fr

Received 17 November 2011; Revised 17 February 2012; Accepted 3 March 2012

Academic Editor: Muhammad Shafique

The canonical signed digit (CSD) representation of constant coefficients is a unique signed data representation containing the fewest number of nonzero bits. Consequently, for constant multipliers, the number of additions and subtractions is minimized by CSD representation of constant coefficients. This technique is mainly used for finite impulse response (FIR) filter by reducing the number of partial products. In this paper, we use CSD with a novel common subexpression elimination (CSE) scheme on the optimal Loeffler algorithm for the computation of discrete cosine transform (DCT). To meet the challenges of low-power and high-speed processing, we present an optimized image compression scheme based on two-dimensional DCT. Finally, a novel and a simple reconfigurable quantization method combined with DCT computation is presented to effectively save the computational complexity. We present here a new DCT architecture based on the proposed technique. From the experimental results obtained from the FPGA prototype we find that the proposed design has several advantages in terms of power reduction, speed performance, and saving of silicon area along with PSNR improvement over the existing designs as well as the Xilinx core.

1. Introduction

Many applications such as video surveillance and patient monitoring systems require many cameras for effective tracking of living and nonliving objects. To manage the huge amount of data generated by several cameras, we proposed an optical implementation of an image compression based on DCT algorithm in [1]. But this solution suffers from bad image quality and higher material complexity. After this optical implementation, in this paper we propose a digital realization of an optimized VLSI for image compression system. This paper is an extension of our prior work [2–4] with a new compression scheme along with supplementary simulations and FPGA implementation followed by performance analysis.

More recent video encoders such as H.263 [5] and MPEG-4 Part 2 [6] use the DCT-based image compression along with additional algorithms for motion estimation (ME). A simplified block diagram of the encoder is presented in Figure 1. The 2D DCT of 8×8 blocks of the image is performed to decorrelate each block of input pixels. The DCT coefficients are then quantized to represent them in a reduced range of values using a quantization matrix. Finally, the quantized components are scanned in a zigzag order, and the encoder employs run-length encoding (RLE) and Huffman coding/binary arithmetic coding (BAC-) based algorithms for entropy coding.

Since the DCT computation and quantization processes are computation intensive, several algorithms are proposed in literature for computing them efficiently in dedicated hardware. Research in this domain can be classified into three parts. The first part is the earliest and concerns the reduction of the number of arithmetic operators required for DCT computation [7–13]. The second research thematic relates to the computation of DCT using multiple constant multiplication schemes [14–25] for hardware implementation. Some other works on design of architectures for DCT make use of convolution formulation. They are efficient but can be used only for prime-length DCT and not suitable for video processing applications [26, 27]. Finally, the third part is about the optimization of the DCT computation in the

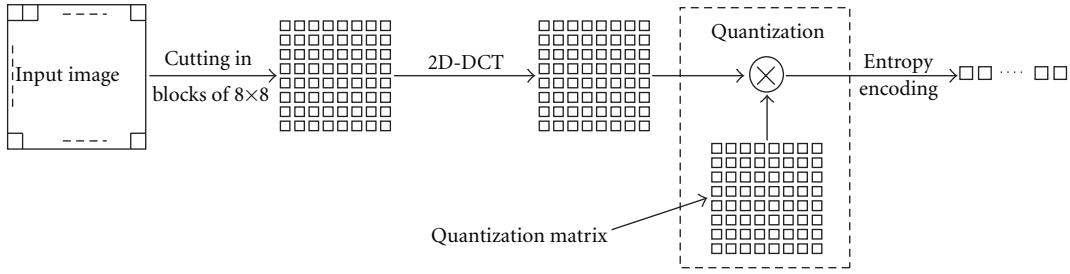

Figure 1: Simplified block diagram of the encoder [2].

context of image and video encoding [28–32]. In this paper, we are interested in the last research thematic.

In this paper, we propose a novel architecture of the DCT based on the canonical signed digit (CSD) encoding [33, 34]. Hartley in [35] has used CSD-based encoding and common subexpression elimination (CSE) for efficient implementation of FIR filter. The use of similar CSD and CSE technique for DCT implementation is not suitable. To improve the efficiency of implementation, we identify multiple subexpression occurrences in intermediate signals (but not in constant coefficients as in [35]) in order to compute DCT outputs. Since the calculation of multiple identical subexpression needs to be implemented only once, the resources necessary for these operations can be shared and the total number of required adders and subtractors can be reduced.

The second contribution of the paper is an introduction of a new schema of image compression where the second stage of 1D DCT (DCT on the columns) is configured for joint optimization of the quantization and the 2D DCT computation. Moreover, tradeoffs between image visual quality, power, silicon area, and computing time are analysed.

The remainder of the paper is organized as follows: an overview of fundamental design issues is given in Section 2. Proposed DCT optimization based on CSD and subexpression sharing is described in Section 3. An algorithm based on joint optimization of quantization and 2D DCT computation is proposed in Section 4. Finally, the experimental results are detailed in the Section 5 before the conclusion.

2. Background

Given an input sequence $\{x(n)\}$, $n \in [0, N-1]$, the N-point DCT is defined as:

$$X(n) = \sqrt{\frac{2}{N}} C(n) \sum_{k=0}^{N-1} x(k) \cos \frac{(2k+1)n\pi}{2N}, \quad (1)$$

where $C(0) = 1/\sqrt{2}$ and $C(n) = 1$ if $n \neq 0$.

As stated in the introduction, we find two main types of algorithms for DCT computation. One class of algorithms is focused on reducing the number of required arithmetic operators, while the other class of algorithms are designed for hardware implementation of DCT. In this Section, we provide a brief review of the major developments of different types of algorithms.

2.1. Fast DCT Algorithm. In literature, many fast DCT algorithms are reported. All of them use the symmetry of the cosine function to reduce the number of multipliers. In [36] a summary of these algorithms is presented. In Table 1, we have listed the number of multipliers and adder involved in different DCT algorithms. In [13], the authors show that the theoretical lower limit of 8-point DCT algorithm is 11 multiplications. Since the number of multiplications of Loeffler's algorithm [12] reaches the theoretical limit, our work is based on this algorithm.

Loeffler et al. in [12] proposed to compute DCT outputs on four stages as shown in Figure 2. The first stage is performed by 4 adders and 4 subtractors while the second one is composed of 2 adders, 2 subtractors, and 2 MultAddSub (multiplier, adder and subtractor) blocks. Each MultAddSub block uses 4 multiplications and can be reduced to 3 multiplications by constant arrangements. The fourth stage uses 2 MultSqrt(2) blocks to perform multiplication by $\sqrt{2}$.

2.2. Multiplierless DCT Architecture. The DCT given by (1) can be expressed in inner product form as:

$$Y = \sum_{k=0}^{N-1} x(k) \cdot c(k), \quad (2)$$

where $c(k)$ for $0 \leq k \leq N-1$ are fixed coefficients and equal to $\cos((2k+1)\pi/2N)$, and $x(k)$ for $0 \leq k \leq N-1$ are the input image pixels.

One possible implementation of the inner product in programmable devices uses embedded multipliers. However, these IPs are not designed for constant multipliers. Consequently, they are not power efficient and consume a larger silicon area. Moreover, such a design is not portable for efficient implementation in FPGAs and ASICs. Many multiplierless architectures have, therefore, been introduced for efficient implementation of constant multiplications for the inner product computation. All those methods can be classified as: the ROM-based design [14], the distributed arithmetic (DA-) based design [15], the New distributed

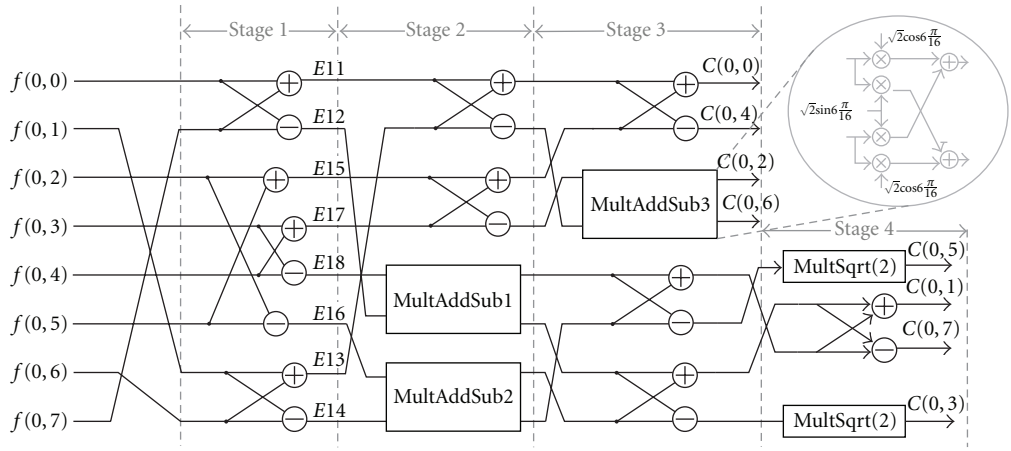

Figure 2: Loeffler architecture of 8-point DCT algorithm.

arithmetic (NEDA-) based design [22], and the CORDIC-based design [23].

2.2.1. ROM Multiplier-Based Implementation. This solution is presented in [14] to design a special-purpose VLSI processor of 8×8 2D DCT/IDCT chip that can be used for high-speed image and video coding. Since the DCT coefficient matrix is fixed, the authors of [14] precompute all possible product values and store them in a ROM rather than computing them by any combinational logic. Since the dynamic range of input pixels is 2^8 for gray scale images, the number of stored values in the ROM is equal to $N = 2^8$. Each value is encoded using 16 bits. For example, for an 8-point inner product, the ROM size is about $8 * 2^8 * 16$ bits which is equivalent to 32.768 kbits. To obtain 8-point DCT, 8-point inner products are required and consequently, the ROM size becomes exorbitant for realization of image compression.

2.2.2. Distributed Arithmetic (DA). Distributed arithmetic (DA) [15] is a well-known technique for computing inner products which outperforms the ROM-based design. Authors of [16–18] use the recursive DCT algorithm to derive a design that requires less area than conventional algorithms. By precomputing all the partial inner products corresponding to all possible bit vectors and storing these values in a ROM, the DA method speeds up the inner product computation over the multiplier-based method. Unfortunately, in this case also the size of ROM grows exponentially with the number of inputs and internal precision. This is inherent to the DA technique where a great amount of redundancy is introduced into the ROM to accommodate all possible combinations of bit patterns in the input signal.

2.2.3. New Distributed Arithmetic (NEDA). The New Distributed Arithmetic (NEDA) is adder-based optimization of DA implementation. The NEDA architecture does not require ROMs and multipliers. It provides reduced complexity solution by sharing of common subexpression of input vector to generate optimal shift-add network for DCT implementation. This results in a low-power, high-throughput architecture for the DCT. Nevertheless, the implementation of NEDA has two main disadvantages, [22]:

(i) the parallel data input leads to higher scanning rate which severely limits the operating frequency of the architecture;

(ii) the assumption of serial data input leads to lower hardware utilization.

2.2.4. CORDIC. COordinate Rotation DIgital Computer (CORDIC) provides a low-cost technique for DCT computation. The CORDIC-based DCT algorithm in [23] utilizes dynamic transformation rather than static ROM addressing. The CORDIC method can be employed in two different modes: the rotation mode and the vectoring mode. Sun et al. in [24] have presented an efficient Loeffler DCT architecture based on the CORDIC algorithm. However, the use of dynamic computation of cosine function in iterative way involves long latency, high-power consumption and involves a costly scale compensation circuit.

2.2.5. CSD. Vinod and lai in [25] have proposed an algorithm to reduce the number of operations by using CSD and CSE techniques, where they have minimized the number of switching events in order to reduce the power consumption. In CSD encoding, the constant multiplications are replaced by additions. Hence, there are two types of additions: interstructural adders to compute the summation terms of the inner product of (2) and intrastructural adder required to replace the constant multipliers. The authors of [25] applied the CSD encoding on the constant multiplier of the conventional DCT computation. Consequently, the number of intrastructural adders is reduced, but the number of interstructural adders is increased to 56 for 8-point DCT. However, as it is reported in Table 1, there are some fast DCT algorithms which use the cosine symmetry to reduce the number of adders (from 26 to 29). Moreover, the authors of [25] have used CSE technique to reduce the number of intrastructural adders. Indeed, since each data (image

Table 1: Complexity of different DCT algorithms.

Reference	[7]	[8]	[9]	[10]	[11]	[12]
Multipliers	16	12	12	12	12	11
Adders	26	29	29	29	29	29

pixel) is multiplied with distinct constant element (cosine coefficient), Vinod and lai have proposed to reformulate the DCT matrix for efficient substitution of CSE. With this optimization, the total number of intrastructural adders substantially reduced.

2.3. Joint Optimization. Recent research on DCT implementation uses the optimization to adapt the implementation of the DCT to the specific compression standards in order to reduce the chip size and power consumption. All these recent works in this area exploit the context in which the DCT is used to reduce the computational complexity. Some of them use the characteristics of input signals and the others simplify the DCT architecture. Xanthopoulos and Chandraksan in [28] have exploited the signal correlation property to design a DCT core with low-power dissipation. Yang and Wang have investigated the joint optimization of the Huffman tables, quantization, and DCT [31]. They have tried to find the performance limit of the JPEG encoder by proposing an iterative algorithm to find the optimal DCT bit width for a given Huffman tables and quantization step sizes.

A prediction algorithm is developed in [32] by Hsu and Cheng to reduce the computation complexity of the DCT and the quantization process of H264 standard. They have built a mathematical model based on the offset of the DCT coefficients to develop a prediction algorithm.

In this paper, we propose a model for combined optimization of DCT and quantization to implement them in the same architecture to save the computational complexity for image and video compression.

3. Proposed Algorithm for DCT Computation

In this Section, we present a new multiplierless DCT based on CSD encoding.

3.1. Principle of CSD. The CSD representation was first introduced by Avizienis in [33] as a signed-digit representation of numbers. This representation was created originally to eliminate the carry propagation chains in arithmetic operations. It is a unique signed-digit representation containing the fewest number of nonzero bits. It is therefore used for the implementation of constant multiplications with the minimum number of additions and subtractions. The CSD representation of any given number c is given by:

$$c = \sum_{i=0}^{N-1} c_i \cdot 2^i, \quad c_i = \{-1, 0, 1\}, \tag{3}$$

CSD numbers have two basic properties:

(i) no two consecutive digits in a CSD number are nonzero;

Table 2: 8-point DCT fixed coefficient representation.

Real value	Decimal	Natural binary	Partial products	CSD	Partial products
$\cos(3\pi/16)$	106	01101010	4	+0−0+0+0	4
$\sin(3\pi/16)$	71	01000111	4	0+00+00−	3
$\cos(\pi/16)$	126	01111110	6	+00000−0	2
$\sin(\pi/16)$	25	00011001	3	00+0−00+	3
$\cos(6\pi/16)$	49	00110001	3	0+0−000+	3
$\sin(6\pi/16)$	118	01110110	5	+000−0−0	3
$\sqrt{(2)}$	181	10110101	5	+0−0−0+0+	5
Total partial products			30		23

(ii) The CSD representation of a number contains the minimum possible number of nonzero bits and thus the name canonic.

The CSD values of the constants used in 8-point DCT by Loeffler algorithm are listed in Table 2. The digits 1,−1 are, respectively, represented by +,−. For 8 bit width, the saving in term of partial products is about 24%. A generalized statistical study about the average number of nonzero elements in N-bit CSD numbers is presented in [33], and it is proved that this number tends asymptotically to $N/3 + 1/9$. Hence, on average, CSD numbers contain about 33% fewer nonzero bits than 2's complement numbers. Consequently, for multiplications by a constant (where the bit pattern is fixed and known a priori), the numbers of partial products are reduced by nearly 33% in average.

3.2. New CSE Technique for DCT Implementation. To minimize the number of adders, subtractors, and shift operators for DCT computation, we can use the common subexpression elimination (CSE) technique over the CSD representation of constants. CSE was introduced in [35] and applied to digital filters in transpose form. Contrary to transpose form FIR filters, constant coefficients of DCT (shown in Table 2) multiply 8 *different input data* since the DCT consists in transforming 8-point input sequence to 8-point output coefficient. For this reason we cannot exploit the redundancy among the constants for subexpression elimination as in case of FIR filter. Moreover, for bit patterns in the same constant, Table 2 shows that only the constant $\sqrt{2}$ presents one common subexpression which is +0− repeated once with an opposite sign. Consequently, we cannot use the conventional CSE technique in the same manner as in the case of multiple constant multiplication in FIR filters.

We have proposed here a new CSE approach for DCT optimization where we do not consider occurrences in CSD coefficients, but we consider the interaction of these codes. On the other hand, according to our compression method (detailed in the next Section) we use only some of the DCT coefficients (1 to 5 among 8). Hence, it is necessary to compute specific outputs separately. To emphasize the advantage of CSE, we take the example of

$X(2)(X(2) = E35 + E37)$. According to Figure 2, we can express $E35$ as follows:

$$\begin{aligned} E35 &= (E25 + E28) \\ &= \left(E18 * \cos\left(\frac{3\pi}{16}\right) + E12 * \sin\left(\frac{3\pi}{16}\right)\right) \\ &\quad + \left(E14 * \cos\left(\frac{\pi}{16}\right) - E16 * \sin\left(\frac{\pi}{16}\right)\right). \end{aligned} \tag{4}$$

Using CSD encoding of Table 2, (7) is equivalent to:

$$\begin{aligned} E35 = {} & E18(2^7 - 2^5 + 2^3 + 2^1) + E12(2^6 + 2^3 - 2^0) \\ & - E16(2^5 - 2^3 + 2^0) + E14(2^7 - 2^1). \end{aligned} \tag{5}$$

After rearrangement (8) is equivalent to:

$$\begin{aligned} E35 = {} & 2^7(E18 + E14) + 2^6E12 - 2^5(E16 + E18) \\ & + 2^3(E12 + E16 + E18) + 2^1(E18 - E14) \\ & - 2^0(E12 + E16). \end{aligned} \tag{6}$$

In the same way, we can determine $E37$:

$$\begin{aligned} E37 = {} & 2^7(E12 + E16) - 2^6E18 + 2^5(E14 - E12) \\ & + 2^3(E12 - E14 - E18) + 2^1(E12 - E16) \\ & + 2^0(E14 + E18). \end{aligned} \tag{7}$$

Equations (10) and (11) give

$$\begin{aligned} X(2) = {} & 2^7\left(\overbrace{(E16 + E18)}^{\text{CS1}} + E12 + E14\right) + 2^6(E12 - E18) \\ & - 2^5\left(\overbrace{(E12 - E14)}^{\text{CS2}} + \overbrace{(E16 + E18)}^{\text{CS1}}\right) \\ & + 2^3\left(\overbrace{(E12 - E14)}^{\text{CS2}} + E12 + E16\right) \\ & + 2^1\left(\overbrace{(E18 - E16)}^{\text{CS3}} + \overbrace{(E12 - E14)}^{\text{CS2}}\right) \\ & + 2^0\left(\overbrace{(E14 - E12)}^{\text{CS2}} + \overbrace{(E18 - E16)}^{\text{CS3}}\right), \end{aligned} \tag{8}$$

where CS1, CS2, and CS3 denote 3 common subexpressions. In fact, the identification of common subexpressions results in significant reduction of hardware and power consumption reductions. For example, CS2 appears 4 times in $X(2)$. This subexpression is implemented only once and resources needed to compute CS2 are shared. An illustration of resources sharing is given in Figure 3.

Table 3: Statistics of X(2) calculation.

Components/methods	Multiplier based	CSD	CSD-CSE
Adders/subtractors	11	23	16
Registers	125	188	119
MULT18x18SIOs	4	0	0
Equivalent no of LUT	305	221	200
Latency	$3T_A + T_M$	$6T_A$	$4T_A$
Maximum frequency[1]	143.451	121.734	165.888

[1]Maximum frequency is measured in MHz.

Symbols $\ll n$ denote left shift operation by n-bit positions. It is important to notice that nonoverbraced terms in (12) are potential common subexpressions which could be shared with other DCT coefficients such as $X(4)$, $X(6)$, and $X(8)$. According to this analysis, $X(2)$ is computed by using 11 adders and 4 embedded multipliers. If CSD encoding, is applied 23 adders/subtractors, are required. The proposed method enables to compute $X(2)$ by using only 16 add/subtract operations. This improvement allows to save silicon area and reduces the power consumption without any decrease in the maximum operating frequency.

To emphasize the common subexpression sharing, a VHDL model of calculation of $X(2)$ is developed using three techniques: embedded multipliers, CSD encoding, and CSE of CSD encoding. It is shown in Table 3 that the CSD encoding uses more adders, subtractors, and registers than the proposed combined CSD-CSE technique to replace the 4 embedded multipliers MULT18x18SIOs. Also, we have included the equivalent number of LUT if $X(2)$ is synthesized on Xilinx FPGA without any arithmetic DSP core (without MULT18x18). Total number of LUTs is found to be 305, 221, and 200, respectively, for the multiplier-based design, the CSD-based design, and the combined CSD-CSE-based design. Moreover, it can be observed that the time required to get $X(2)$ coefficient is equal to $3T_A + T_M$, $6T_A$, and $4T_A$, respectively, for the multiplier-based design, the CSD-based design, and the combined CSD-CSE-based design, where T_A is the addition time and T_M is the multiplication time. Consequently, the area-delay product is decreased by sharing subexpression.

4. Joint Optimization

4.1. Principle. As discussed earlier, the 2D DCT is computed in two stages by row/column decomposition using row-wise 1D DCT of input in stage 1, followed by column-wise 1D DCT of intermediate result in stage 2. If we consider an input block $f(i, j)$ of 8×8 samples, the row-wise transform calculates $X(i, v)$ as 1D DCT of $f(i, j)$, and the column-wise transform gives $Y(u, v)$ which are the 1D DCT coefficients applied to $X(i, v)$ for $i, j, u, v \in [1 : 8]$. Hence, for a given 8×8 block of pixels, we obtain 64 DCT coefficients of different frequencies. Unlike the high-frequency coefficients, the low-frequency coefficients have a greater effect on image reconstruction. Moreover, after quantization process, most of the high-frequency coefficients are likely to be zero as

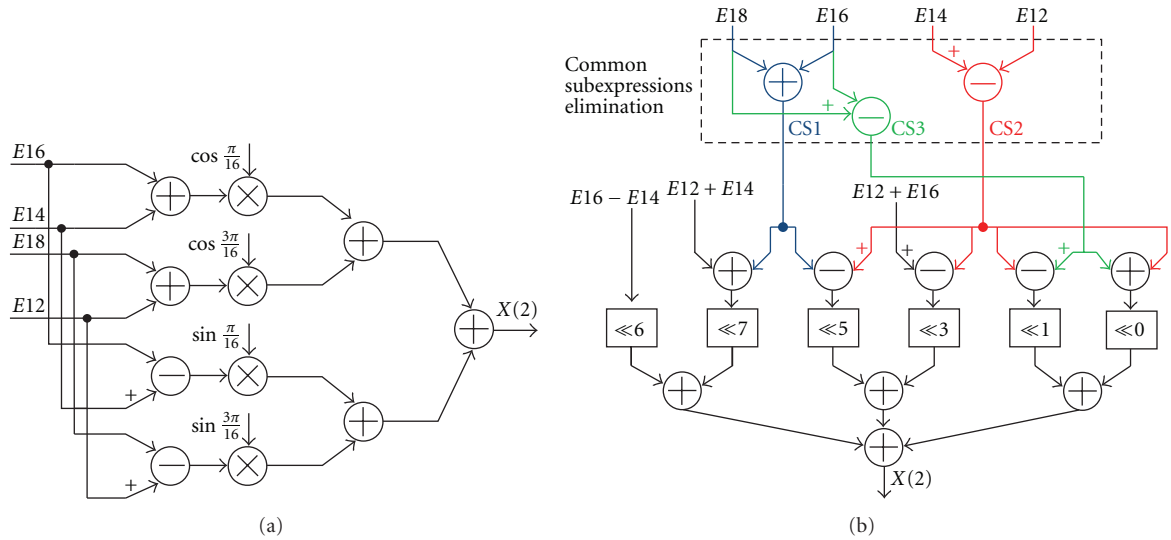

FIGURE 3: $X(2)$ calculation (a) conventional method, (b) shared subexpression using CSD encoding.

shown in Figure 4. Since these coefficients are likely to be zero after quantization, to save computation time and resources we avoid computing these DCT coefficients. In fact, for an 8×8 block of pixels, we compute 64 1D DCT coefficients of the first stage and then we compute only low frequency components for 1D DCT of second stage. With this method, the quantization is done on the fly with the DCT algorithm and consequently we save computational resources. Another advantage of the proposed method is the latency improvement. Since the high-frequency coefficients are to be eventually discarded, the time used for their computation is saved. In fact, for the second 1D DCT algorithm, at least 3 rows do not need to be computed as illustrated in the Figure 5. This gives a saving of at least 12 clock cycles, since the latency of calculation of each row is 4 clock cycles.

4.2. Quantization Levels. For an 8×8 block of pixels, the 1D DCT is calculated for each of the 8 input rows as mentioned in Figure 5. For each row, the first 1D DCT coefficient $X(i, 1)$ is encoded using 11 bits which is the estimated word length without truncation or rounding for $i \in [1 : 8]$. Coefficients $X(i, 2)$ to $X(i, 8)$ are truncated using 8 bits to trade accuracy for compression ratio and computational complexity.

In the second stage, 1D DCT is calculated selectively since the higher frequency components need not be computed. It is known that the lower frequencies tend to spread across either the first row or the first column of the 2D DCT coefficient matrix. However, the computation of an entire row and entire column leads to the computation of all DCT coefficients. For this purpose, we propose an efficient and simple computing scheme by creating 4 DCT zones (shown in Figure 5) where each zone corresponds to a specific compression ratio. For an 8×8 block of pixels, the row-wise transform is applied to compute 64 DCT coefficients while the column-wise transform is applied partially to reduce the computational complexity. Indeed, the proposed quantization zones are chosen to be square in order to avoid redundancy of computation. In Zone 1, only 4 coefficients $Y(1, 1)$, $Y(1, 2)$, $Y(2, 1)$, and $Y(2, 2)$ are calculated by 2 1D DCT operations. The first one is applied to the first column of intermediate result (output of the row-wise transform) which gives $Y(1, 1)$ and $Y(1, 2)$ while the second one is applied to the second column of intermediate result to compute $Y(2, 1)$ and $Y(2, 2)$. For this quantization mode, all the others DCT coefficients are set to zero. Similarly, in Zone 4, 25 DCT coefficients are calculated by 5 1D DCT operations to compute coefficients $Y(u, v)$, $u, v \in [1 : 5]$.

The compression ratio depends on the zone selection. In Zone 1, $Y(1, 1)$ is encoded using 14 bits. Indeed, the first 1D-DCT coefficient is encoded using 11 bits since in Loeffler DCT algorithm (shown in Figure 2), the DC output is obtained by three cascaded adder stages applied to 8-bit image pixels. Since the 11-bit DC output is fed to the second stage of 1D DCT the bit width of the first output of 2D-DCT is equal to 14 bits. This bit width is taken as reference for encoding the AC coefficients which have less influence than DC coefficient on the quality of reconstructed images. To estimate the bit width of AC coefficients, we were referred to image and video quantization tables (Q) in JPEG standard for Luminance image component and in MPEG-4 standard for intraframe video coding. It is found that for image quality of 50%, $Y(2, 2)$ is divided by $Q(2, 2) = 24$. Then, in order to have a unique bit width by quantization zone, AC coefficients of zone 1 are encoded with 5 bits under the DC bit width (i.e., 9 bits). Likewise, AC coefficients of zone 2 need coefficients of zone 1 along with 5 other coefficients $Y(1, 3)$, $Y(2, 3)$, $Y(3, 1)$, $Y(3, 2)$, and $Y(3, 3)$. All these data are encoded using 8 bits. The additional coefficients of zone 3 are encoded using 7 bits. Finally, the remaining coefficients of zone 4 are encoded using 6 bits since $Y(5, 5)$ will be divided by $Q(5, 5) = 136$.

Hence, by selecting zone 1, the total number of bits is equal to $14 + 9 * 3 = 41$ bits and the compression ratio (CR)

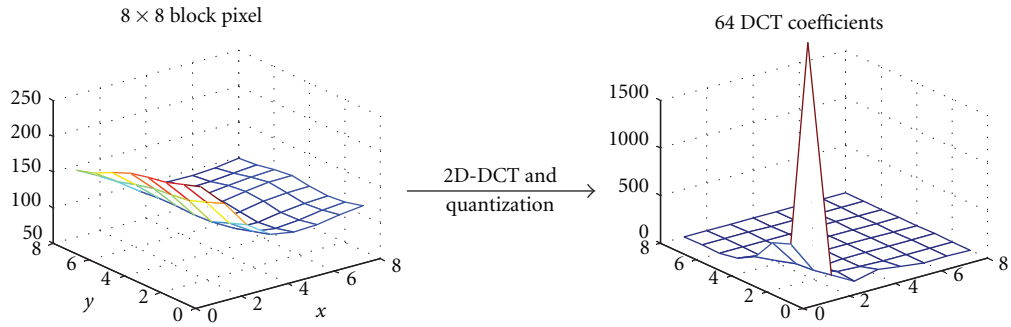

Figure 4: Principle of the DCT and the quantization.

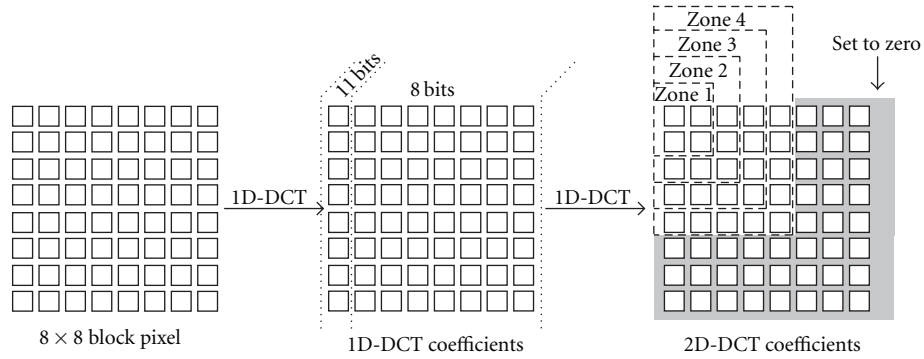

Figure 5: Quantization zones.

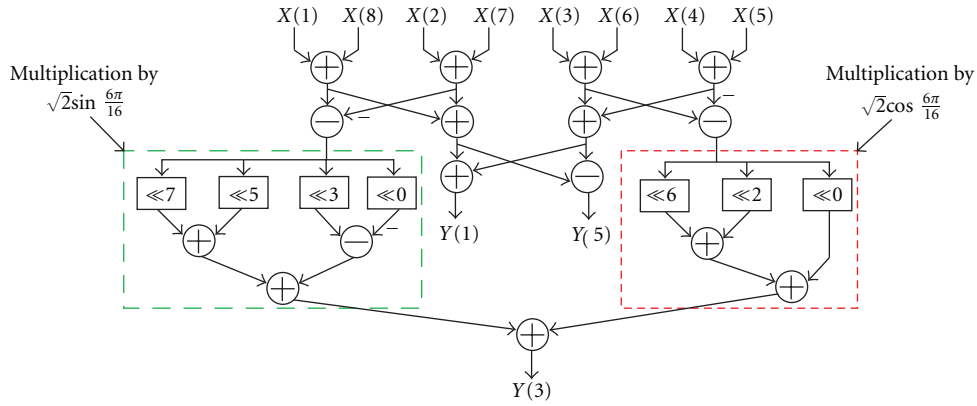

Figure 6: $Y(1,v)$, $Y(3,v)$, and $Y(5,v)$ calculation with CSE and CSD.

which is the ratio of the original image size to the compressed image size is equal to 8 bits $*$ 64/14 bits = 12.48. For the zone 2, the compression ratio is equal to 8 bits $*$ 64/(14 + 9 $*$ 3 + 8 $*$ 5 bits) = 6.32. Similarly, for zone 3 and 4, the CRs are, respectively, equal to 3.95 and 2.78.

4.3. DCT Calculation. An example of the calculation of the 1D DCT coefficient ($X(i,2)$ for $i \in [1:8]$) is given in Section 3.2. To compute the DCT coefficients of the 1D DCT of second stage, we use the same method by replacing the inputs by $X(i,v)$ and the outputs by $Y(u,v)$ for $u,v \in [1:8]$. According to the algorithm illustrated in Figure 2, for a given column, $Y(1,v)$ and $Y(5,v)$ are calculated using adders and subtractors while $Y(3,v)$ uses a multiplicative constant and is given by:

$$Y(3) = \sqrt{2}\cos\left(\frac{6\pi}{16}\right)E24 + \sqrt{2}\sin\left(\frac{6\pi}{16}\right)E22. \quad (9)$$

Intermediate results $E22$ and $E24$ in (9) are shown in Figure 2 for a given column. Now, constants $\sqrt{2}\cos(6\pi/16)$ and $\sqrt{2}\sin(6\pi/16)$ are converted to CSD format and given, respectively, by 0 + 000 + 0+ and 0 + 0 + 0 + 00−. $Y(1,v)$, $Y(3,v)$ and $Y(5,v)$ calculations are given in Figure 6.

For $Y(2,v)$ and $Y(4,v)$ also the CSD-CSE techniques are used. $Y(2,v)$ is calculated as in (8), and the common subexpression of $Y(4,v)$ calculation is determined by increasing

the number of common subexpressions shared between $Y(2,v)$ and $Y(4,v)$.

According to Figure 2, $Y(4,v) = \sqrt{2}(E26 - E27)$ which is equivalent to:

$$\begin{aligned} Y(4,v) = {} & \sqrt{2}\left(E12 * \cos\left(\frac{3\pi}{16}\right) - E18 * \sin\left(\frac{3\pi}{16}\right)\right) \\ & - \sqrt{2}\left(E16 * \cos\left(\frac{\pi}{16}\right) - E14 * \sin\left(\frac{\pi}{16}\right)\right). \end{aligned} \tag{10}$$

Using CSD encoding of the constant coefficient, (10) is equivalent to:

$$\begin{aligned} Y(4,v) = {} & E12(2^7 + 2^5 - 2^3 - 2^0) \\ & - E18(2^7 - 2^5 + 2^2 + 2^0) \\ & - E16(2^8 - 2^6 - 2^4 + 2^1) \\ & + E14(2^5 + 2^2 - 2^0). \end{aligned} \tag{11}$$

After rearrangement (11) is equivalent to:

$$\begin{aligned} Y(4,v) = {} & 2^7\left(-\overbrace{(E16+E18)}^{\text{CS1}} + \overbrace{(E12-E16)}^{\text{CS4}}\right) \\ & + 2^5\left(-\overbrace{(E16+E18)}^{\text{CS1}} + \overbrace{(E12+E14)}^{\text{CS5}} + E14\right) \\ & - 2^3\left(\overbrace{(E12-E16)}^{\text{CS4}}\right) \\ & + 2^2\left(E16 + E14 - \overbrace{(E18-E16)}^{\text{CS3}}\right) \\ & - 2^0\left(\overbrace{(E16+E18)}^{\text{CS1}} + \overbrace{(E12+E14)}^{\text{CS5}} + E16\right), \end{aligned} \tag{12}$$

For $Y(4,v)$ calculation, the common subexpression CS1 and CS3 defined for $Y(2,v)$ calculation is used. Two new subexpressions CS4 and CS5 are introduced to further reduce the arithmetic operators. It is important to mention that the equations listed before are expressed to create several occurrences of common subexpression such as CS1, CS3, and CS4 those are used for $Y(2,v)$ calculation. The Signal flow graphs of $Y(2,v)$ and $Y(4,v)$ are shown in Figure 7.

5. Simulation Results

We have coded the proposed method and the existing competing algorithms in VHDL and synthesized them using Xilinx ISE tool.

Table 4: Macrostatistics of 1D DCT calculation.

Method	DA [16]	DA [18]	NEDA [19]	CSD [25]	Proposed
Adders	136	144	85	123	72

Table 5: Microstatistics of 1D DCT calculation.

Method	NEDA [19]	CORDIC	[37]	Xilinx's core	Proposed
Slices	1031	780	531	369[2]	454

[2] Apart from the number 369 slices Xilinx core uses 4 embedded multipliers.

5.1. Synthesis Results. From high-level synthesis results we obtain the number of adders used for different DA-based 1D DCT design and listed in Table 4. It is found that our design uses fewer adders than the other. The direct realization of DA-based DCT design requires 308 adders. Optimizations presented in [19] reduce the number of adders to 85. Regarding the CSD-based design [25], for 8-bit constant width, we found that design of [25] consumes 123 adders (67 intrastrucutral adders + 56 interstructural adders) while the proposed design involves the DCT with 72 adders. We have listed the number of slices occupied by the 1D DCT of [37] and proposed design in the Table 5. The proposed method is compared favorably with the conventional multiplierless architectures using Xilinx XC2VP50 FPGA, the same device employed in [37].

Note that the Xilinx's core uses the Chen's algorithm [7] and requires 369 Slices along with 4 embedded multipliers 18x18SIOs. Besides, the number of slices required by the Xilinx core is relatively low compared with other designs because, the adder/subtractor module of the Xilinx's design alternatively chooses addition and subtraction by using a toggle flop. However, the slice-delay product of proposed design is significantly less than that of the Xilinx DCT IP core since the later has a maximum usable frequency (MUF) of 101 MHz on Spartan3E device while the proposed design provides MUF of 119 MHz.

Regarding the timing analysis derived from synthesis results obtained by cadence 0.18 μ library, we find that the proposed design involves a delay about 14.4 ns which represents nearly 15% less than the Xilinx core and 60% less than optimized NEDA-based design [20]. We should underline that in [20] an optimized architecture of NEDA-based design [19] where compressor trees are used to decrease the delay.

Moreover, we have used the XPower tool of Xilinx ISE suite to estimate the dynamic power consumption. The power dissipation of the proposed 1D DCT design and Xilinx's core is about 39 mW and 62 mW respectively.

In order to highlight the effect of subexpression sharing, 1D DCT structure of Loeffler algorithm is implemented with different multiplier designs. The power-delay product in nJ is computed as the product of the DCT computation time (ns) and the power dissipation (W) for the proposed design and the Xilinx core. The power-delay product for different number of DCT coefficients is calculated and plotted in Figure 8. The CSD-based design and the proposed design using CSE involve nearly 43% and 33% of power-delay

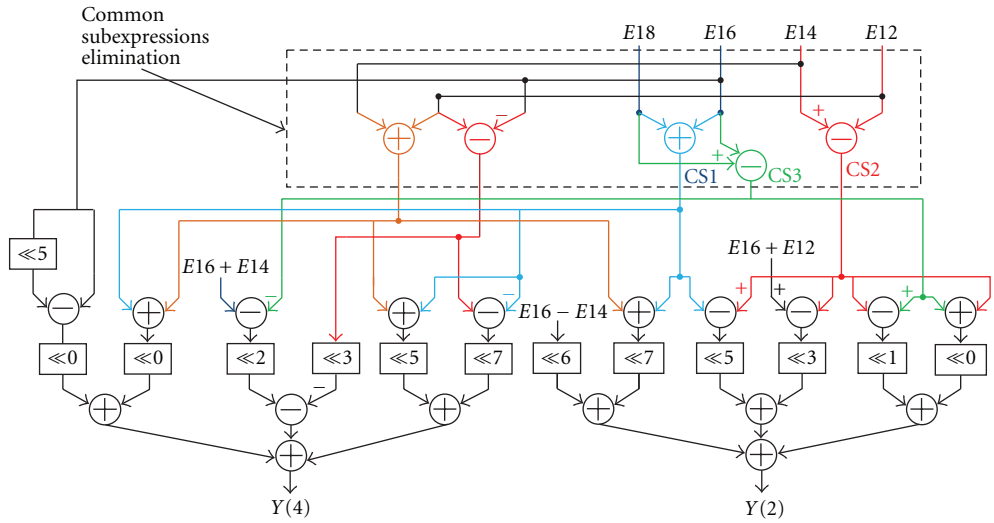

Figure 7: $Y(2, v)$ and $Y(4, v)$ calculation with CSE and CSD.

product of the Xilinx's multiplier-based design, respectively. It can be seen in Figure 8 that the computation of only the first 1D DCT coefficient involves the same power-delay product since this coefficient does not require any multiplier. Note that the computation of 4th DCT coefficient requires nearly the same power-delay product as that for 5th DCT coefficient. Indeed, the computation of the fifth DCT coefficient requires only one more subtrator.

For the 2D DCT architecture using the Loeffler algorithm a performance analysis is presented in Table 6 in order to highlight the effects of CSD coding, subexpression sharing, and quantization. The multiplier-based structures considered in the comparison are the Xilinx's embedded multiplier synthesized as multiplier block IP and in LUTs. It should be indicated that the input bit width is of 8 bits, the DC coefficient bit width of the first and second 1D DCT stages are 11 bits and 14 bits, respectively and the constant cosine coefficient bit width is 8 bits. The implementation of 2D DCT is realized by decomposing the 2D DCT into two 1D DCT computations together with a transpose memory. It can be observed in Table 6 that the area-delay complexity of Xinlinx's multiplier-based 2D DCT design (synthesized in block) is nearly the same as that of the combined CSD-CSE design but has nearly twice the power consumption. On the other hand, when the Xilinx's multipliers are synthesized as LUT, the 2D DCT structure has less power-delay product but involves twice the area compared with the combined CSD-CSE structure.

The average computation time (ACT) is the time interval after which we get a set of 2D DCT coefficients. ACT is the product of the number of clock cycles required for the 2D DCT computation and the duration of a clock cycle. The 2D DCT computation requires 86 cycles, which is comprised of 8 cycles for register inputs, 7 cycles for the first stage 1D DCT, 64 cycles for transpose memory, and 7 cycles for the second stage of 1D DCT.

Finally, we use the energy per output coefficient (EoC) as power metric which amounts to the average of energy

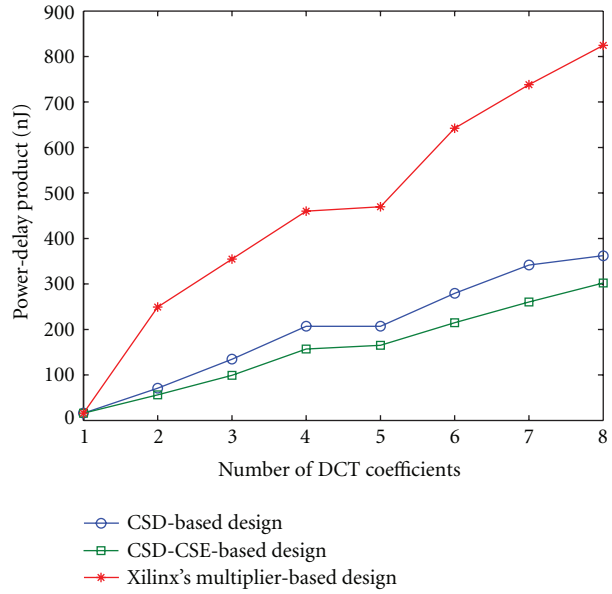

Figure 8: Power-delay product estimation with 1.6 V design.

required to compute one value of 2D DCT output. EoC is calculated by multiplying the ACT by the power consumption and dividing the product by 64. It is shown in Table 6 that the design based on Xilinx's multiplier IP involves more than twice EoC compared with all the other designs. Moreover, the proposed 2D DCT with CSD-CSE technique needs less energy compared with CSD-based design and Xilinx's multiplier-based design. It can be further observed that the proposed CSD-CSE and quantization technique has 46% to 65% of EoC compared to CSD-based design.

5.2. FPGA Implementation of Image Compression. In this subsection, we examine the quality of reconstructed image using an FPGA prototype of the proposed DCT-based image

Table 6: Performance analysis of the 2D DCT design using Loeffler algorithm.

Constant multiplication design	Slice	MULT18x18SIOs	Power dissipation (mW)	Delay (ns)	ACT (μs)	EoC (nJ)
Xilinx-embedded multipliers (Block)	960	20	176	20.58	1.769	311.344
Xilinx-embedded multipliers (LUT)	1836	0	66.4	24.02	2.065	137.116
CSD	1423	0	74	22.39	1.925	142.45
CSD-CSE	1048	0	76	19.94	1.714	130.264
CSD-CSE and quantization	[625, 765]	0	[48, 64]	[15.90, 16.89]	[1.367, 1.452]	[65.616, 92.928]

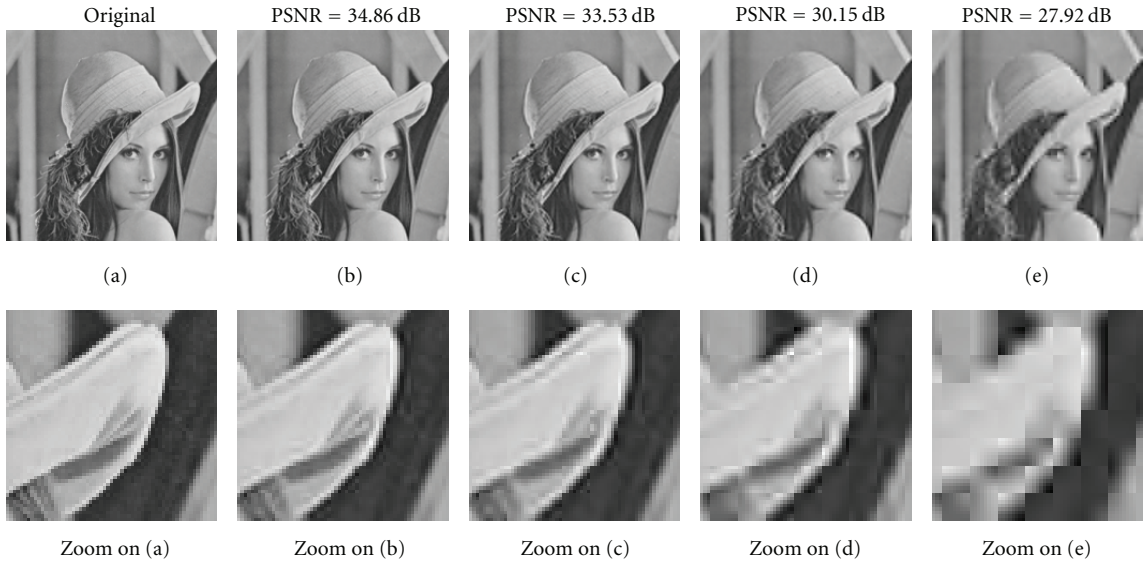

Figure 9: Decoded images after joint optimization. (a) (Original), (b) (quantization zone = 4, bpp = 2.87, PSNR = 33.24 dB), (c) (quantization zone = 3, bpp = 2.0, PSNR = 30.26 dB), (d) (quantization zone = 2, bpp =1.26, PSNR = 28.23 dB), and (e) (quantization zone = 1, bpp=0.64, PSNR = 25.38 dB).

Table 7: PSNR (dB) versus bpp evaluation.

bpp	0.64	1.26	2.0	2.87
Lena	25.38	28.23	30.26	33.24
Mandrill	20.84	25.27	28.20	29.36
Peppers	27.92	30.15	33.53	34.86
Goldhill	27.51	29.22	32.43	33.12

compression unit. The test images are saved in a ROM in order to avoid the transmission time between the PC and the FPGA. It is important to mention that in the final design we need to use a 2-bit word to indicate 4 available compression ratios. To measure the visual quality of the reconstructed image and to validate the proposed DCT design, we use the Xilinx's integrated logic analyzer (ILA). This module works as a digital oscilloscope and enables to trigger on signals in the hardware design.

To reconstruct back the images, a floating point inverse 2D DCT function of Matlab tool is applied to the FPGA output. PSNR of different 255 $\times$ 255 gray scale images are evaluated and listed in Table 7. The bit per pixel (bpp) depends on the quantization zone selection and varies from 0.64 to 2.87 (The compression is due to DCT only. To increase the compression ratio further the quantized DCT output needs to pass through the entropy coding which we have not performed here.). As shown in Table 7, a good or acceptable image visual qualities can be obtained by joint optimization of the quantization and the DCT. Moreover, to underline the adequacy between PSNR results and the user perception, in Figure 9 we have shown the decoded images for different selection of quantization zones. It is found that the higher the PSNR of reconstructed image, the better the quality is.

6. Conclusion

In this paper, we have presented a low-complexity DCT-based image compression. We presented a novel common subexpression sharing of intermediate signals of the DCT computation based on CSD representation of Loeffler's 8-point DCT algorithm. Finally, we have combined the quantization process with the second stage of DCT computation

in order to optimize the bit width of computation of DCT coefficient according to the quantization of different zones.

We would like to point out that a prior detection of zero-quantized coefficients along with the proposed techniques could be used to further reduce the complexity of DCT computations.

References

[1] A. Alkholidi, A. Alfalou, and H. Hamam, "A new approach for optical colored image compression using the JPEG standards," *Signal Processing*, vol. 87, no. 4, pp. 569–583, 2007.

[2] M. Jridi and A. Alfalou, "Joint Optimization of Low-power DCT Architecture and Effcient Quantization Technique for Embedded Image Compression," in *VLSI-SoC: Forward-Looking Trends in IC and System Design*, J. Ayala, A. Alonso, and R. Reis, Eds., pp. 155–181, Springer, Berlin, Germany, 2012.

[3] M. Jridi and A. Alfalou, "A low-power, high-speed DCT architecture for image compression: Principle and implementation," in *18th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC '10)*, pp. 304–309, September 2010.

[4] M. Jridi and A. AlFalou, "A VLSI implementation of a new simultaneous images compression and encryption method," in *IEEE International Conference on Imaging Systems and Techniques (IST '10)*, pp. 75–79, July 2010.

[5] "Video coding for low bit rate communication," (ITU-T Rec. H.263), February 1998.

[6] ISO/IEC DIS 10 918-1, "Coding of audio visual objects: part 2. visual," ISO/IEC 14496-2 (MPEG-4 Part2), January 1999.

[7] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004–1009, 1977.

[8] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1243–1245, 1984.

[9] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, vol. 6, no. 4, pp. 267–278, 1984.

[10] N. Suehiro and M. Hatori, "Fast algorithms for DFT and other sinusoidal tranforms," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, pp. 642–664, 1986.

[11] H. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, pp. 1455–1461, 1987.

[12] C. Loeffler, A. Lightenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithm with 11 multiplications," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP '89)*, pp. 988–991, May 1989.

[13] P. Duhamel and H. H'mida, "New 2n DCT algorithm suitable for VLSI implementation," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP '87)*, pp. 1805–1808, November 1987.

[14] D. Slawecki and W. Li, "DCT/IDCT processor design for high data rate image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 2, pp. 135–146, 1992.

[15] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 4–19, 1989.

[16] A. Madisetti and A. N. Willson, "100 MHz 2-D 8 × 8 DCT/IDCT processor for HDTV applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 2, pp. 158–165, 1995.

[17] S. Yu and E. E. Swartzlander, "DCT implementation with distributed arithmetic," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 985–991, 2001.

[18] D. W. Kim, T. W. Kwon, J. M. Seo et al., "A compatible DCT/IDCT architecture using hardwired distributed arithmatic," in *IEEE International Symposium on Circuits and Systems (ISCAS '01)*, pp. 457–460, May 2001.

[19] A. Shams, W. Pan, A. Chidanandan, and M. Bayoumi, "A low power high performance distributed DCT architecture," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '02)*, pp. 21–27, 2002.

[20] A. Chidanandan, J. Moder, and M. Bayoumi, "Implementation of NEDA-based DCT architecture using even-odd decomposition of the 8 × 8 DCT matrix," in *49th Midwest Symposium on Circuits and Systems (MWSCAS '06)*, pp. 600–603, August 2007.

[21] P. K. Meher, "Unified systolic-like architecture for DCT and DST using distributed arithmetic," *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 12, pp. 2656–2663, 2006.

[22] M. Alam, W. Badawy, and G. Jullien, "A new time distributed DCT architecture for MPEG-4 hardware reference model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 5, pp. 726–730, 2005.

[23] S. Yu and E. E. Swartzlander, "A scaled DCT architecture with the CORDIC algorithm," *IEEE Transactions on Signal Processing*, vol. 50, no. 1, pp. 160–167, 2002.

[24] C. C. Sun, S. J. Ruan, B. Heyne, and J. Goetze, "Low-power and high-quality Cordic-based Loeffler DCT for signal processing," *IET Circuits, Devices and Systems*, vol. 1, no. 6, pp. 453–461, 2007.

[25] A. P. Vinod and E. M. K. Lai, "Hardware efficient DCT implementation for portable multimedia terminals using subexpression sharing," in *IEEE Region 10 Annual International Conference (TENCON '04)*, pp. A227–A230, November 2004.

[26] C. Cheng and K. K. Parhi, "A novel systolic array structure for DCT," *IEEE Transactions on Circuits and Systems II*, vol. 52, no. 7, pp. 366–369, 2005.

[27] P. K. Meher, "Systolic designs for DCT using a low-complexity concurrent convolutional formulation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 9, pp. 1041–1050, 2006.

[28] T. Xanthopoulos and A. P. Chandrakasan, "A low-power dct core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 5, pp. 740–750, 2000.

[29] J. Huang and J. Lee, "A self-reconfigurable platform for scalable dct computation using compressed partial bitstreams and blockram prefetching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 11, pp. 1623–1632, 2009.

[30] J. Huang and J. Lee, "Efficient VLSI architecture for video transcoding," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1462–1470, 2009.

[31] E. H. Yang and L. Wang, "Joint optimization of run-length coding, Huffman coding, and quantization table with complete baseline JPEG decoder compatibility," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 63–74, 2009.

[32] C. L. Hsu and C. H. Cheng, "Reduction of discrete cosine transform/quantisation/inverse quantisation/inverse discrete cosine transform computational complexity in H.264 video encoding by using an efficient prediction algorithm," *IET Image Processing*, vol. 3, no. 4, pp. 177–187, 2009.

[33] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Transaction on Electronic Computers*, vol. 10, pp. 389–400, 1961.

[34] R. H. Seegal, "The canonical signed digit code structure for FIR filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 5, pp. 590–592, 1980.

[35] R. T. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Transactions on Circuits and Systems II*, vol. 43, no. 10, pp. 677–688, 1996.

[36] C. Y. Pai, W. E. Lynch, and A. J. Al-Khalili, "Low-power data-dependent 8 × 8 DCT/IDCT for video compression," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 150, no. 4, pp. 245–255, 2003.

[37] B. I. Kim and S. G. Ziavras, "Low-power multiplierless DCT for image/video coders," in *13th International Symposium on Consumer Electronics (ISCE '09)*, pp. 133–136, May 2009.