Joint Optimization of Low-Power DCT Architecture and Efficient Quantization Technique for Embedded Image Compression

Maher Jridi and Ayman Alfalou

L@bISen Laboratory, Equipe Vision
ISEN-Brest CS 42807 Brest, France
maher.jridi@isen.fr
http://www.isen.fr/brest

Abstract. The Discrete Cosine Transform (DCT)-based image compression is widely used in today's communication systems. Significant research devoted to this domain has demonstrated that the optical compression methods can offer a higher speed but suffer from bad image quality and a growing complexity. To meet the challenges of higher image quality and high speed processing, in this chapter, we present a joint system for DCT-based image compression by combining a VLSI architecture of the DCT algorithm and an efficient quantization technique. Our approach is, firstly, based on a new granularity method in order to take advantage of the adjacent pixel correlation of the input blocks and to improve the visual quality of the reconstructed image. Second, a new architecture based on the Canonical Signed Digit and a novel Common Subexpression Elimination technique is proposed to replace the constant multipliers. Finally, a reconfigurable quantization method is presented to effectively save the computational complexity. Experimental results obtained with a prototype based on FPGA implementation and comparisons with existing works corroborate the validity of the proposed optimizations in terms of power reduction, speed increase, silicon area saving and PSNR improvement.

Keywords: Embedded Image and video compression, Digital hardware implementation, VLSI, Granularity analysis, Multiplierless DCT, Canonical Signed Digit.

1 Introduction

Today, the needs for new processing, transmissions and communications tools have increased significantly to support the extraordinary development of modern telecommunications systems. For fast multimedia communication systems it becomes urgent to compress the target images. At source, as the image is in optical form, many researchers and groups have proposed and validated new optical compression methods [1]. The use of optics is also motivated by the very fast computing time with which the optic allows to process an image. In addition, it is possible to combine the optical compression step with the encryption step

J.L. Ayala, D. Atienza, and R. Reis (Eds.): VLSI-SoC 2010, IFIP AICT 373, pp. 155–181, 2012.

needed to keep the intellectual and/or private property for a target image [2]. One possibility to achieve these optical methods consists in the processing in the frequency domain [1]. To obtain this frequency domain a transformation of the image plane is needed; one of these used transformations is the Discrete Cosine Transform DCT.

We have implemented optically and validated some image compression methods based on the DCT algorithm [3]. But this solution suffers from bad reconstructed image quality and higher material complexity due to the use of Spatial Light Modulator (SLM), optical components. After this optical implementation of an adapted JPEG model, we are interested in the implementation of a simultaneous compression and encryption system [3] and [4].

With all these optical set-ups implementing several architectures, we have concluded that an all optical implementation is not easily possible to replace all the potential of digital processing. In addition, for complex applications, an all optical processing is not enough to have a reliable system. Therefore, an optical processing should be only a part of the system.

On the other hand, reconfigurable hardware, such us FPGAs, are considered as an attractive solution for signal and image processing implementation due to their high-speed I/Os, embedded memories and to their capability to efficiently implement highly parallelized architecture. In 2011, FPGAs under industrialization (Xilinx Series 7) use the 28 nm technology, 2 million logic cells with clock speeds of up to 600 MHz. The new trends on the manufacturing process consists in using Intellectual Properties (IP). And, the goal behind this is to reduce dramatically the Time To Market. However, the dedicated architecture of IP hard cores constitutes a limitation for the implementation of image coding algorithms such as JPEG, MPEG and H26x. In fact, these standards only normalize the algorithm and the decoding format. The method of encoding is left free to competition, as long as the image produced is decoded by a standard decoder [5]. Consequently, flexibility and adaptive computing capabilities are needed to enhance the encoding efficiency and to meet the real-time requirement. It becomes contradictory to design flexible and computational intensive systems with fixed architecture of IP hard cores without an increase in consumed power and silicon area. According to this point of view, we are interested in this chapter on the gate level description of an optimized digital realization of an image compression scheme.

Indeed, one of the most popular image compression scheme is the JPEG coding system largely used in World-Wide-Web and in digital cameras. More recent video encoders such as H.263 [6] and MPEG-4 Part 2 [7] use the same JPEG structure for compression with additional algorithms such as Motion Estimator (ME). This means that the proposed architectures in this chapter may be easily adapted for video encoding.

A simplified block diagram of the JPEG encoder is presented in Fig.1. The first operation consists in dividing the input image into several 8x8 pixel blocks. Then, the 2D-DCT algorithm is applied to decorrelate each block of input pixels. The calculated DCT coefficients are quantized to represent them in a

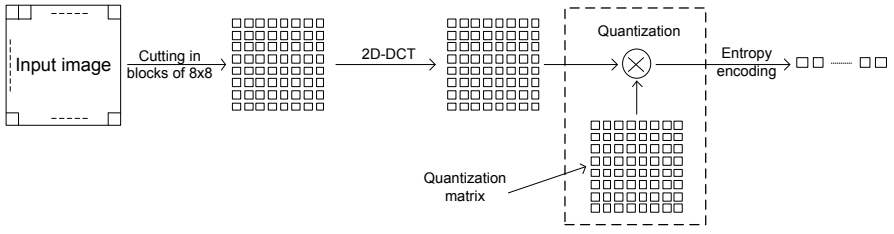


Fig. 1. Simplified block diagram of the JPEG encoder

reduced range of values using a quantization matrix. Finally, the JPEG standard arranges the quantized components in a zigzag order and then employs the Run-Length Encoding (RLE) algorithm that groups similar frequencies together and then uses Huffman coding or arithmetic coding as entropy encoders. Since the DCT algorithm and quantization process are computation-intensive, several improvements are proposed in literature for computing them efficiently. Significant research work has been devoted to the problems of DCT and quantization complexity. These works (analyzed in section II) can be classified into three parts. The first part is the earliest and concerns the reduction of the number of required arithmetic operators of the DCT algorithm [8–14]. The second research thematic is related to the complexity reduction of the constant multipliers used in the DCT. Typically, ROM-based design [15], Distributed Arithmetic (DA) architecture [16–21], the New Distributed Arithmetic (NEDA) [22], the LUT-based design [23] and CORDIC-based design [24, 25] are the most interesting improvements for the multiplierless DCT. Finally, the third part is about the joint optimization of the DCT and the quantization for scalable and reconfigurable image and video encoder [27–30].

In this chapter, we propose three contributions.

1. First, we present an efficient granularity of input pixels to take advantage of the adjacent pixel correlation and to reduce the word length of the multiplier input and consequently the roundoff error. However, we have to underline that the proposed granularity needs a small changes in standard decoders to reorder the generated code stream.
2. Then, we propose a novel architecture of the DCT based on the Canonical Signed Digit (CSD) encoding based on our recent work of [31]. In fact, the use of subtractor with adders and shift operators allows an efficient implementation [32, 33]. Hartley in [34] identify common elements in CSD constant coefficients of FIR filter and share required resources. The latter is named Common Subexpression Elimination (CSE). The use of this technique, with the same manner as in FIR filter, is studied in this paper but the results are not satisfying. To overcome this problem we propose to identify multiple subexpression occurrences in intermediate signals (but not in constant coefficients as in [34]) 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. The total number of required adders and subtractors can be reduced.

3. Finally, we introduce a simple reconfigurable scheme of image compression based on the joint optimization between the DCT computation and the quantization process. We demonstrate by computing some specific DCT coefficients that the visual quality of reconstructed images is very close to the quality obtained with encoder standard which use matrix multiplications. A tradeoffs between image visual quality, power, silicon area and computing time is made.

This paper is organized as follows: an overview of fundamental design issues is given in section 2. Analysis of granularity of input pixel is presented in Section 3. DCT optimization based on CSD and subexpression sharing is described in section 4. Then, a joint optimization of quantization and DCT algorithm is proposed in section 5 along with experimental results before the conclusion.

2 DCT Optimization Techniques

2.1 Choice of the Algorithm

In this section the DCT algorithm is reviewed. Given an input sequence $x(n)$, $n \in [0, N-1]$, the N-point DCT is defined as:

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

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

As stated in the introduction, DCT optimization has focused first on reducing the number of required arithmetic operators. 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 [35] a summary of these algorithms was presented. Table 1 sums up these results.

In [14], the authors have showed that the theoretical lower limit of 8-point DCT algorithm is 11 multiplications. Since the number of multiplications of Loeffler's algorithm [13] reaches the theoretical limit, our work is based on this algorithm.

Table 1. Complexity of different DCT algorithms

Reference	Chen [8]	Lee [9]	Vitterli [10]	Suehiro [11]	Hou [12]	Loeffler [13]
Multipliers	16	12	12	12	12	11
Adders	26	29	29	29	29	29

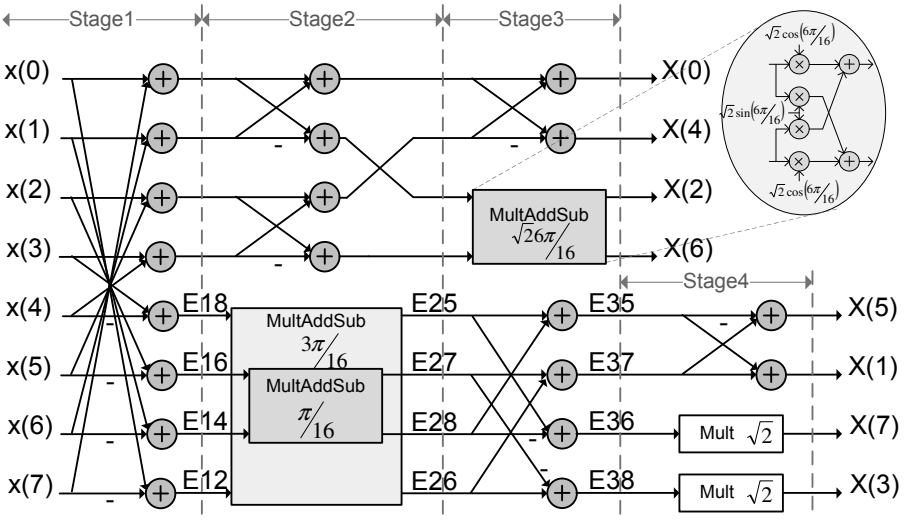


Fig. 2. Loeffler architecture of 8-point DCT algorithm

Loeffler in [13] has proposed to compute DCT by using four stages as shown in Fig. 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 Mult blocks to compute the multiplication by $\sqrt{2}$.

2.2 Multiplierless DCT Architecture

The multipliers are used in DCT to solve the equation of inner product detailed in (2):

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

where $c\,(k)$ are fixed coefficients and equal to $cos\frac{(2k+1)\pi}{2N}$ and $x\,(k)$ are the input image pixels.

One possible implementation of the inner product consists in using embedded multipliers of the FPGA. However, embedded multipliers are not designed for constant multipliers. Consequently, they are not power efficient and consume a large silicon area. Moreover, the obtained design is not portable to all FPGA families and ASICs since we use specified IP hard cores. Many multiplierless architectures have, therefore, been introduced for efficient computation of the inner product in DSP applications. The ROM-based design [15], the Distributed

Arithmetic (DA) [16, 18], the LUT-based computing [23], the New Distributed Arithmetic (NEDA) [22] and the CORDIC [24] are the most popular architectures.

ROM-Based Multiplier. The ROM-based multiplier computing is close to human-like computing. This solution is presented in [15] to design a special-purpose VLSI processors of 8x8 2-D DCT/IDCT chip that can be used for high rate image and video coding. In the case of DCT, the coefficient matrix is constant; hence the authors of [15] precomputed all the multiplier outputs and store them in a ROM rather than compute these values on line. As 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 8-point inner products are required and consequently, the ROM size becomes exorbitant especially for image compression.

Distributed Arithmetic (DA). Authors of [18] use the recursive DCT algorithm and their design requires less area than conventional algorithms. The design of [18] consider the same inner product in (2) and rewrite the variable $x(k)$ as follows:

$$x(k) = \sum_{b=0}^{B-1} x_b(k).2^b \qquad x_b(k) = \{0, 1\} \tag{3}$$

where $x_b(k)$ is the b^{th} bit of $x(k)$ and B is the resolution of the input vector. Finally, the inner product can be rewritten as follows:

$$Y = \sum_{k=0}^{N-1} c(k) \sum_{b=0}^{B-1} x_b(k).2^b \tag{4}$$

The equation (4) can be rearranged as follows:

$$Y = \sum_{b=0}^{B-1} \left[\sum_{k=0}^{N-1} c(k).x_b(k) \right] .2^b \tag{5}$$

Equation (5) defines distributed arithmetic computation. In fact, the bracketed term in (5) may have only 2^N possible values since $x_b(k)$ may take on values of 0 and 1 only. Since $c(k)$ are fixed coefficient values, we can compute the bracketed term in (5) by storing 2^N possible combinations in a ROM. Input data can be used to address the ROM. Now, for the computation of the inner product, we can use an accumulator with a shift operator as mentioned in the Fig. 3. The result of the inner product is available after N clock cycles. By precomputing all the possible values and storing these values in a ROM, the DA method speeds

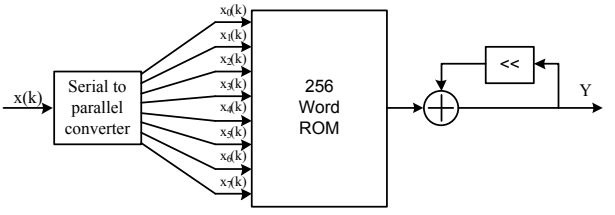


Fig. 3. Distributed Arithmetic principle

up the multiply process. Unfortunately, the size of ROM grows exponentially when the number of inputs and internal precision increase. This is inherent to the DA mechanism where a great amount of redundancy is introduced into the ROM to accommodate all possible combinations of bit patterns exhibited by the input signal.

New Distributed Arithmetic (NEDA). The New Distributed Arithmetic (NEDA) is based on the optimization of DA architecture. The bits of the constant DCT coefficients are distributed to perform the DCT operation with just addition operations. The NEDA architecture is without ROMs and multipliers. This results in a low power, high throughput architecture for the DCT. Nevertheless, the implementation of NEDA has two main disadvantages, [22]:

- the parallel data input leads to higher scanning rate which will severely limit the operating frequency of the architecture;
- the assumption of serial data input leads to lower hardware utilization.

LUT-Based Multiplier. A recent work on LUT-based multiplier computing [23] employs three optimization techniques for inner-product calculation used in many DSP applications such as FIR filters, convolutions and sinusoidal transforms. The first technique is the Odd-Multiple Storage (OMS) scheme which consists on storing only the odd multiple of the constant, the even multiples could be derived from the stored words. The second technique is the Anti-symmetric Product Coding (APC) scheme which involves only half the number of product words are to be saved. The last technique is the Input Coding (IC) scheme where the input word x is decomposed into certain number of segments or sub-words $x = (x_1, x_2, ..., x_T)$ and fed to separate LUTs.

Obtained performances with the LUT-based multiplier are at least similar to DA approach for the same throughput.

CORDIC. COordinate Rotation DIgital Computer (CORDIC) is a very interesting technique for phase to sine amplitude conversion. This algorithm proposed in [24] 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. In the rotation mode, the algorithm basic idea consists in decomposing rotation operation into successive basic rotations. Each basic rotation can be realized by shifting and adding arithmetic operations to compute the sine and cosine functions. A detailed architecture of the CORDIC algorithm applied to digital synthesizer can be found in [26].

C.-C. Sun et al. in [25] presented an efficient Loeffler DCT architecture based on the CORDIC algorithm. However, the use of dynamic computation of cosine function in iterative way is time consuming (long latency), power consuming and requires a complicated compensation method.

To implement the constant multiplication a new solution is proposed in section IV. Comparison results with NEDA and CORDIC are provided.

2.3 Joint Optimization

One way of optimizing DCT computation consists in computing DCT coefficients jointly with other algorithms used in the compression scheme. Many work about the DCT joint optimization adapt the implementation of the DCT to the compression standards and reduce the material requirement and the power consumption. Some of them use the statistic behavior of DCT input signals to yield with the DCT coefficients and the others simplify the DCT architecture by introducing the quantization operation, the motion estimation and compensation algorithms or also the entropy encoding algorithm.

Xanthopoulos and Chandraksan in [27] exploited the signal correlation to design a DCT core. They used an MSB rejection module to reduce the number of arithmetic operations. Their chip allows the user to program statically the maximum desired precision due to quantization.

Huang and Lee in [29] proposed an efficient video transcoder architecture. To design the motion estimation and compensation, they used the DCT statistical properties.

Yang and Wang investigated in [5] the joint optimization of the Huffman tables, quantization and DCT. They tried to find the performance limit of the JPEG encoder by presenting an iterative algorithm to find the optimal DCT coefficients for a given Huffman tables and quantization step sizes.

A prediction algorithm is developed in [30] by Hsu and Cheng to reduce the computation complexity of the DCT and the quantization process of the H264 standard. They built a mathematical model based on the offset of the DCT algorithm to develop a prediction algorithm to save the computational complexity of the video encoder components.

3 Granularity Analysis

3.1 Principle

The most common way to implement the 2D-DCT algorithm is the row/column decomposition. The row/column approach consists of two 1D-DCTs algorithm and one transposed memory to realize a 2D-DCT chip. All the realizations of

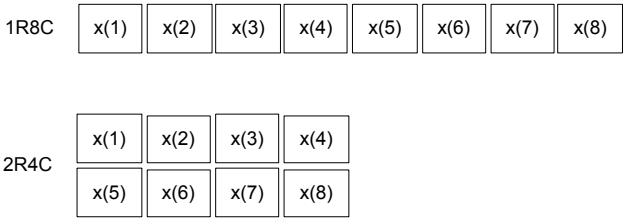


Fig. 4. Granularities distribution

1D-DCT use 8 pixels which constitute a good compromise between the consumed material resources and the operating frequency. In this section we are interested to the 1D-DCT optimization which may be used in our dedicated image compression scheme based on 1D-DCT or 2D-DCT algorithms. This means that the proposed design of 1D-DCT can be easily extended for 2D algorithm.

For a given 8x8 pixel block, the 8 input pixels for the DCT are always taken in the same row and on 8 consecutive columns. We give the name of 1R8C for this configuration. Therefore, we propose to study a non-standard-compliant granularity using 2 consecutive rows and 4 consecutive columns to form the 8 input pixels. We give the name of 2R4C for this proposed granularity. These granularities are presented in Fig. 4.

The advantage of the second granularity compared to the first one relies on the fact that the input pixels have a strong dependency since the image used are often natural and highly structured images. Thus, the pixels of 2R4C granularity are, in general, more correlated than those of 1R8C granularity. This remark is very interesting in the case of signals at the output of subtractors in the first stage of the 1D-DCT algorithm in Fig. 2. In fact, in the case of 2R4C, signals E12, E14 and E16 use more correlated inputs and consequently the dynamic ranges of these signal are reduced. Hence, two solutions are possible. The first one consists in encoding these signals with fewer bits and consequently the multiplier requirements of the stage 2 of the 1D-DCT are reduced. The second solution consists in keeping constant the length of these signals in order to minimize the roundoff error of arithmetic operators in the following stages (the signal truncating is done in the last stage). This second solution is adopted in this paper in order to obtain a high image quality. For the multiplier requirements, a solution will be given in the section 4.

3.2 Verification

To verify the advantage of the proposed granularity compared to the 1R8C, we define for a given row of 8 pixels (in the case of 1R8C), $dx_k(j)$ as the difference between the intensities of two pixels spaced by $k-1$ pixels as:

$$dx_k(j) = |x(j) - x(j+k)| \tag{6}$$

Table 2. Intensity distributions of th first stage subtraction outputs

Signal	Expression	1R8C	2R4C
E12	$x(1) - x(8)$	dx_7	$dx_3 + dy$
E14	$x(2) - x(7)$	dx_5	$dx_1 + dy$
E16	$x(3) - x(6)$	dx_3	$dx_1 + dy$
E18	$x(4) - x(5)$	dx_1	$dx_3 + dy$

where $k \in [1, 7]$ and j is the pixel index, $j \in [1, 8-k]$. For example, if $k = 7$, $j = 1$ and $dx_7\,(1)$ calculates the absolute difference between the intensities of the first and the eighth pixel. For the 2R4C granularity, we define the same $dx_k\,(j)$ where $k \in [1, 3]$ and $j \in [1, 4-k]$. Furthermore, we define $dy\,(i)$ as the absolute difference between two adjacent vertical pixels as:

$$dy\,(i) = |x\,(i) - x\,(i+4)| \tag{7}$$

with $i \in [1, 4]$.

Without loss of generality, we assume that absolute difference between two adjacent vertical pixels do not depend on the position of the pixels in the block. Consequently, we suppose that $dy\,(i)$, for $i \in [1, 4]$, have the same distribution in terms of standard deviation and dynamic range. Hereafter, we use dy for the absolute difference between two adjacent vertical pixels. In the same manner, we use dx_k for the absolute difference between the intensities of two pixels spaced by $k-1$ pixels. Table 2 uses dy and dx_k to determine the intensity distribution of the DCT first stage subtraction outputs using 1R8C and 2R4C granularities. For E12, E14 and E16 signals, the dynamic range of the absolute difference of pixel intensity is reduced in the case of 2R4C.

3.3 Validation

To show the advantage of the 2R4C granularity, a statistical study is performed. The Table 3 shows the standard deviations of intermediate signals E12, E14, E16, E18 using six 256x256 standard gray scaled images. We can notice for the first five images that the granularity 2R4C is more interesting than 1R8C since the standard deviation is smaller for the E12, E14, E16 signals. Moreover, the dynamic range of these signals is smaller in the case of 2R4C. This has a great impact on the visual quality of the reconstructed image. In fact, all arithmetic operators have a fixed-point arithmetic because it is not possible to keep an infinite resolution of operators. Consequently, a roundoff noise due to arithmetic rounding operation is created. It is obvious that, for a fixed point operation, the magnitude of the roundoff noise is proportional to the input dynamic range. Consequently, the 2R4C granularity is more interesting than 1R8C one in terms of reconstructed image quality.

Table 3. Standard deviations of intermediate signals

Image	E12		E14		E16		E18	
	2R4C	1R8C	2R4C	1R8C	2R4C	1R8C	2R4C	1R8C
Lena	33.46	46.67	21.73	40.81	19.02	32.30	31.89	19.88
Barbara	28.02	53.22	16.80	50.26	15.36	40.27	26.55	21.84
Mandrill	40.81	44.75	37.31	43.37	38.46	40.50	40.89	32.26
Peppers	46.88	65.38	28.91	51.68	27.74	39.26	46.11	19.24
Finger-point	85.83	95.23	67.7804	107.18	75.22	112.43	86.55	59.86
Horizontal texture	8.48	0.18	8.48	0.17	8.47	0.15	8.47	0.11

To emphasize this effect, two compression and decompression models are proposed using the 1R8C and the 2R4C granularities to form 8 input pixels for the 1D-DCT. For these two configurations, a 1D-DCT fixed point (FxP) Loeffler algorithm is used. To reconstruct the images, 1D-Inverse DCT (1D-IDCT) floating point (FlP) function of Matlab tool is employed. In these models the quantization step is bypassed in order to evaluate the difference between the granularities for any given bit rate. A synoptic diagram of these models is shown in Fig. 5 and used to compare the two configurations.

The Peak Signal to Noise Ratio (PSNR) is used as criteria to compare the visual quality of the reconstructed images compared with original images. The evaluation of the PSNR is illustrated in Fig. 6. According to these results, for a given DCT output length, an effective gain of PSNR is achieved with the proposed granularity. For a DCT output length of 10 bits, the 2R4C granularity presents about 7 dB and 4 dB more than the 1R8C granularity respectively for Lena and Mandrill 256x256 gray scale images.

According to these results, for a given DCT output length, an effective gain of PSNR is achieved with the proposed granularity. For a DCT output length of 10 bits, the 2R4C granularity presents about 7 dB and 4 dB more than the 1R8C granularity respectively for Lena and Mandrill 256x256 gray scale images.

3.4 Critical Analysis

As stated before, the proposed 2R4C granularity presents more correlation between input pixels and consequently the intermediate signals E12, E14 and E16 have a smaller standard deviation and dynamic range. Conversely, signal E18 presents a different behavior. Despite this, we still have gain in PSNR because signal E18 contributes equally with E12, E14 and E16 to compute the DCT outputs.

On the other side, there is a second critical point needing more details. In fact, we can see in Table 3 that the image called horizontal texture presents a smaller standard deviation in the case of 1R8C. Hence, for these kind of images the 1R8C

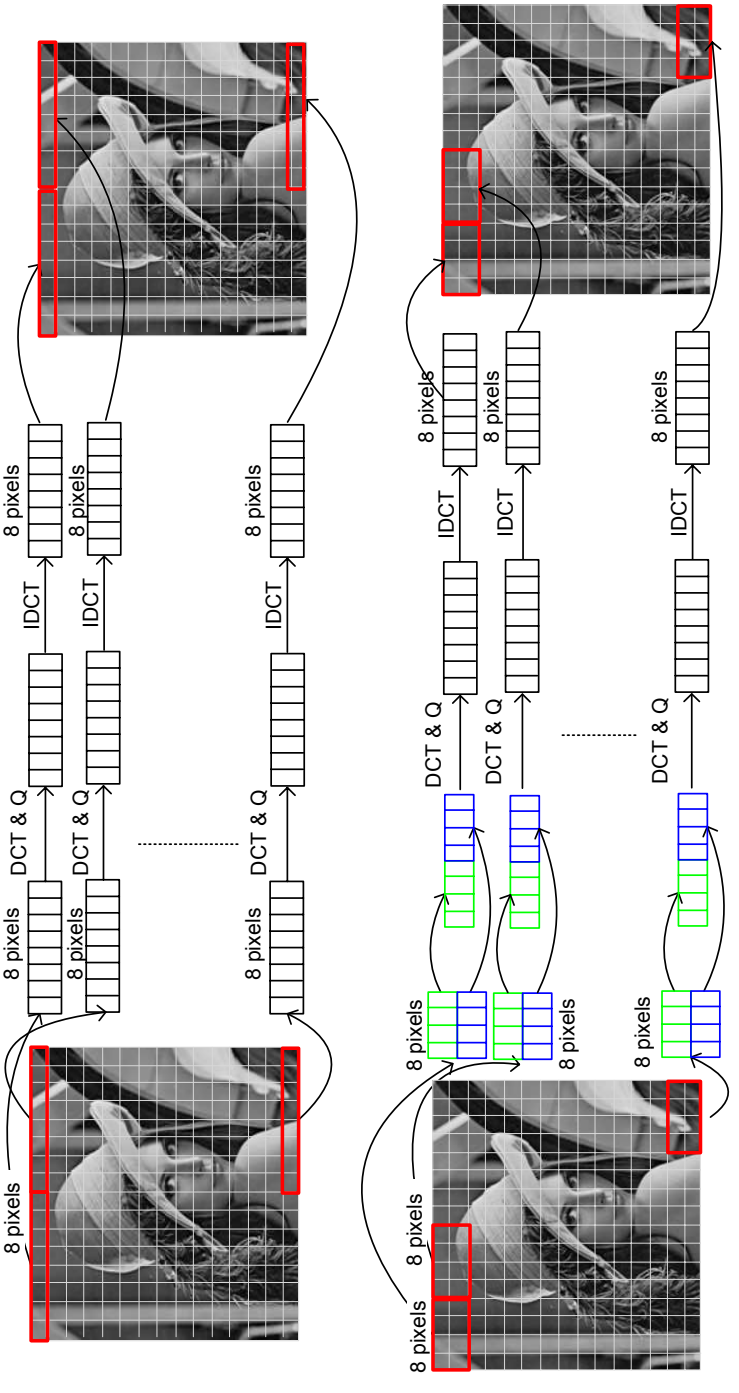


Fig. 5. Synoptic diagram of the compression model based on 1D-DCT and 1D-IDCT

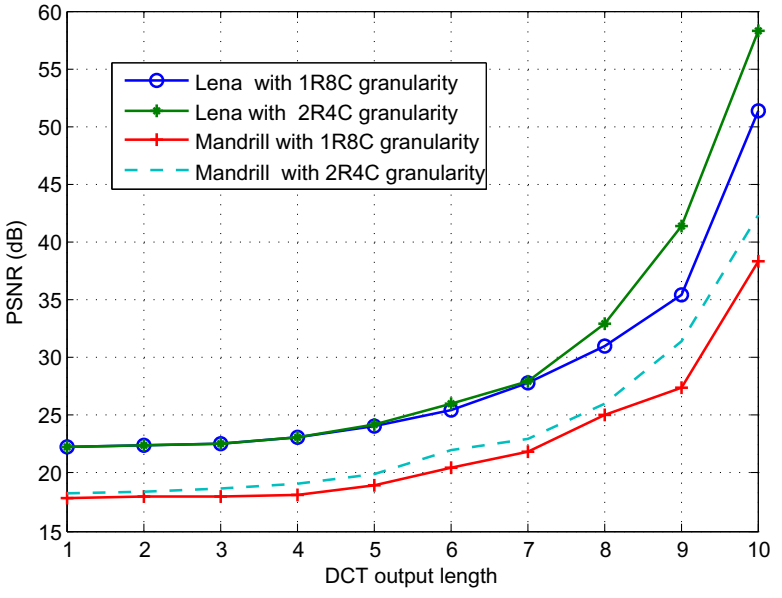


Fig. 6. Simulation results of the compression models

is better. But, we have to notice that for this image the standard deviation of 2R4C granularity is small compared to other standard images. Consequently, the size of arithmetic operators in stage 2 of 1D-DCT in Fig. 2 stills small.

4 A New Power-Efficient DCT

In this section a new multiplierless and romless low-power DCT based in the CSD encoding is presented.

4.1 Principle

The CSD representation was first introduced by Avizienis in [32] as a signed representation. This data representation was created originally to eliminate the carry propagation chains in arithmetic operations. The CSD representation is a unique signed data representation containing the fewest number of nonzero bits. Consequently, for constant multipliers, the number of additions and subtractions will be minimum.

In fact, for a constant coefficient c, the CSD representation is expressed as follows:

$$c = \sum_{i=0}^{N-1} c_i.2^i \qquad c_i = \{-1, 0, 1\} \tag{8}$$

CSD numbers have essentially two properties:

Table 4. Cosine coefficients representation of 8-point DCT

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

– No 2 consecutive bits in a CSD number are nonzero;
– The CSD representation of a number contains the minimum possible number of nonzero bits, thus the name canonic.

Examples of conversion between natural binary representation and CSD representation are given in Tables 4 and 5 where the converted values are the constants used in 8-point and 16-point DCT Loeffler algorithm respectively. Symbols 1,-1 are respectively represented by +,-. For high order DCT such as 16-point DCT, the CSD representation of cosine coefficients is mentioned in Table 5. In the line 4 of Table 4, $128 * cos(\pi/16) \approx 126$ contains 6 partial products since its binary representation is 01111110. In CSD convention, 126 is represented by +00000-0 which is $+2^7 - 2^1$. An extended study of CSD encoding applied to 16-point DCT algorithm is made. The number of partial products is 71 for binary representation and 52 for CSD representation (26% saving). Finally, a generalized statistical study about the average of nonzero elements in N-bit CSD numbers is presented in [32] and prove that this number tends asymptotically to N/3 + 1/9. Hence, on average, CSD numbers contain about 33% fewer nonzero bits than two complement numbers. Therefore, for constant multipliers, the numbers of partial products are reduced and consequently the speed and power could be enhanced.

4.2 New CSE Technique for DCT Algorithm

To enhance the use of adders, subtractors and shift operators we propose to employ the Common Subexpression Elimination (CSE) technique. CSE was introduced in [34] and applied to digital filters. For FIR filters, this technique uses bit pattern occurrence in filter coefficients. Identification of occurrences permits to create subexpression and after that economize hardware resources. Bit pattern occurrence can be detected between two or more different filter coefficients or in the same filter coefficient. Since several constant coefficients multiply only

Table 5. Cosine coefficients representation of 16-point DCT

Real value	Decimal	Natural binary	Partial products	CSD	Partial products
$cos\frac{-15\pi}{32}$	13	00001101	3	000+0-0+	3
$sin\frac{-15\pi}{32}$	-127	01111111	7	+000000-	2
$cos\frac{3\pi}{32}$	122	01111010	5	+000-0+0	3
$sin\frac{3\pi}{32}$	37	00100101	3	00+00+0+	3
$cos\frac{-11\pi}{32}$	60	00111100	4	0+000-00	2
$sin\frac{-11\pi}{32}$	-113	01110001	4	+00-000+	3
$cos\frac{7\pi}{32}$	99	01100011	4	+0-00+0-	4
$sin\frac{7\pi}{32}$	81	01010001	3	0+0+000+	3
$cos\frac{10\pi}{32}$	71	01000111	4	0+00+00-	3
$sin\frac{10\pi}{32}$	106	01101010	4	+0-0+0+0	4
$cos\frac{14\pi}{32}$	25	00011001	3	00+0-00+	3
$sin\frac{14\pi}{32}$	126	01111110	6	+00000-0	2
$cos\frac{12\pi}{32}$	49	00110001	3	0+0-000+	3
$sin\frac{12\pi}{32}$	118	01110110	5	+000-0-0	3
$cos\frac{-12\pi}{32}$	49	00110001	3	0+0-000+	3
$sin\frac{-12\pi}{32}$	-118	01110110	5	+000-0-0	3
$\sqrt{2}$	181	10110101	5	+0-0-0+0+	5
Total Partial products		71		52	

one output data (signal to be filtered), CSE of filter coefficients implies less computation and less power consumption.

Contrary to FIR filters, constant coefficients of DCT in Table 4 multiply 8 different input data since the DCT consists in transforming 8 points in the input to 8 points in the output. From this observation, we can not exploit the redundancy between different constants. Moreover, for bit patterns in the same constant, Table 4 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 CSE technique in the same manner as in the FIR filters.

In order to take advantage of CSE technique, we adapt the CSE for DCT optimization. Proceeding towards this goal, 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 subsection 5.1) we will use only some DCT outputs (1 to 8 among 8). Hence, it is necessary to compute specific outputs separately. To emphasize the CSE effect, we take the example of $X(2)$ and $X(4)$ calculation. In fact, the computation of $X(8)$ and $X(6)$ may be determined from $X(2)$ and $X(4)$ respectively by using the same computation techniques detailed in this subsection. The computations of $X(3)$

and $X\,(7)$ require multiplication but we can apply only the CSD encoding rather that the binary encoding. And, the computation of $X\,(1)$ and $X\,(5)$ coefficients do not need multiplications.

***X* (2) Calculation.** According to Fig. 2, we can express $X\,(2)$ as follows:

$$X(2) = (E35 + E37) \tag{9}$$

And, E35 is expressed by

$$\begin{aligned} E35 &= (E25 + E28) \\ &= (E18 * cos\,(3\pi/16) + E12 * sin\,(3\pi/16)) \\ &+ (E14 * cos\,(\pi/16) - E16 * sin\,(\pi/16)) \end{aligned} \tag{10}$$

Using CSD encoding of Table 4, (10) is equivalent to:

$$\begin{aligned} E35 = E18\left(2^7 - 2^5 + 2^3 + 2^1\right) + E12\left(2^6 + 2^3 - 2^0\right) \\ - E16\left(2^5 - 2^3 + 2^0\right) + E14\left(2^7 - 2^1\right) \end{aligned} \tag{11}$$

After rearrangement (11) is equivalent to:

$$\begin{aligned} E35 = 2^7(E18 + E14) + 2^6E12 - 2^5(E16 + E18) \\ + 2^3(E12 + E16 + E18) + 2^1(E18 - E14) - 2^0(E12 + E16) \end{aligned} \tag{12}$$

In the same way, we can determine E37:

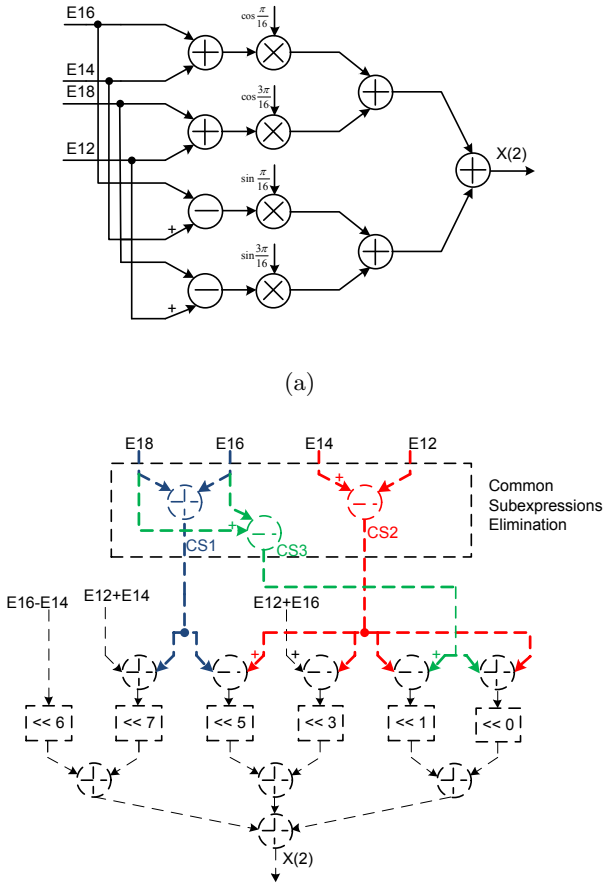
$$\begin{aligned} E37 &= (E26 + E27) \\ &= (E18 * cos\,(3\pi/16) - E12 * sin\,(3\pi/16)) \\ &+ (E16 * cos\,(\pi/16) + E14 * sin\,(\pi/16)) \end{aligned} \tag{13}$$

After CSD encoding (13) gives

$$\begin{aligned} E37 = 2^7(E12 + E16) - 2^6E18 + 2^5(E14 - E12) \\ + 2^3(E12 - E14 - E18) + 2^1(E12 - E16) + 2^0(E14 + E18) \end{aligned} \tag{14}$$

Equations (12) and (14) give:

$$\begin{aligned} X(2) = 2^7(\overbrace{(E16 + E18)}^{CS1} + E12 + E14) + 2^6(E12 - E18) \\ - 2^5(\overbrace{(E12 - E14)}^{CS2} + \overbrace{(E16 + E18)}^{CS1}) + 2^3(\overbrace{(E12 - E14)}^{CS2} + E12 + E16) \\ + 2^1(\overbrace{(E18 - E16)}^{CS3} + \overbrace{(E12 - E14)}^{CS2}) + 2^0(\overbrace{(E14 - E12)}^{CS2} + \overbrace{(E18 - E16)}^{CS3}) \end{aligned} \tag{15}$$



(a)

(b)

Fig. 7. X(2) calculation (a): Conventional method, (b): Shared subexpression using CSD encoding [31]

where CS1, CS2 and CS3 denote 3 common subexpressions. In fact, the identification of common subexpressions can give an important 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 Fig. 7.

Symbols $<< n$ denote right shift operators by n positions. It is important to notice that non-overbraced terms in (15) are a potential common subexpressions with other DCT coefficients such us X(4), X(6) and X(8).

According to this analysis, $X(2)$ is computed by using 11 adders and 4 embedded multipliers or 23 adders and subtractors if the CSD encoding is applied to the constant cosine values. The proposed contribution enables to compute $X(2)$

Table 6. Macro Statistics of X(2) calculation

	Multiplier	CSD	CSD&CSE
Adders/Subtractors	11	23	16
Registers	125	188	119
MULT18X18SIOs	4	0	0
Maximum Frequency (MHz)	143.451	121.734	165.888

by using only 16 adders and subtractors. This improvement allows to save silicon area and reduces the power consumption without any decrease in the operating frequency.

To emphasize the common element sharing, a VHDL model of X(2) calculation is developed using three techniques: embedded multipliers, CSD encoding and CSE of CSD encoding. Results in terms of number arithmetic operators and maximum operating frequency are illustrated in Table 6. It is shown in column 3 that the CSD encoding uses more adders, subtractors and registers to replace the 4 embedded multipliers MULT18x18SIOs (Xilinx's embedded multipliers for Spartan3 family). This substitution between embedded multipliers and arithmetic operators is paid by a loss in the maximum operating frequency. On the other side, column 4 shows that the sharing of arithmetic operators permits to increase the maximum operating frequency and reduce the number of required adders, subtractors and registers.

***X* (4) Calculation.** According to the Fig. 2, one can determine $X(4)$ by

$$X(4) = \sqrt{2}\,(E26 - E27) \tag{16}$$

Equation (16) is equivalent to

$$\begin{aligned} X(4) = {} & \sqrt{2}\,(E12 * cos\,(3\pi/16) - E18 * sin\,(3\pi/16)) \\ & - \sqrt{2}\,(E16 * cos\,(\pi/16) - E14 * sin\,(\pi/16)) \end{aligned} \tag{17}$$

Using CSD encoding of the constant coefficient, (17) is equivalent to:

$$\begin{aligned} X(4) = {} & E12\left(2^7 + 2^5 - 2^3 - 2^0\right) - E18\left(2^7 - 2^5 + 2^2 + 2^0\right) \\ & - E16\left(2^8 - 2^6 - 2^4 + 2^1\right) + E14\left(2^5 + 2^2 - 2^0\right) \end{aligned} \tag{18}$$

After rearrangement (18) is equivalent to:

$$\begin{aligned} X(4) = {} & 2^7(-\overbrace{(E16 + E18)}^{CS1} + \overbrace{(E12 - E16)}^{CS4}) + 2^5(-\overbrace{(E16 + E18)}^{CS1} \\ & + \overbrace{(E12 + E14)}^{CS5} + E14) - 2^3(\overbrace{(E12 - E16)}^{CS4}) + 2^2(E16 + E14 \\ & - \overbrace{(E18 - E16)}^{CS3}) - 2^0(\overbrace{(E16 + E18)}^{CS1} + \overbrace{(E12 + E14)}^{CS5} + E16) \end{aligned} \tag{19}$$

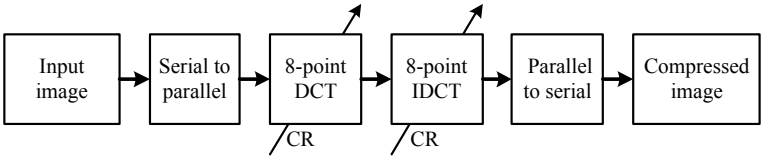


Fig. 8. Model of the image compression technique

Hence, for X (4) calculation, the common subexpression CS1 and CS3 defined for X (2) calculation are used. Two new subexpressions CS4 and CS5 are introduced to economize more 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 also CS4 which is used for X (2) calculation.

5 Validation for Image Compression

The compression technique used in this work is inspired from [3] and the associated model is illustrated in Fig. 8. This compression was realized using optical components to take benefits of speed. Indeed, this kind of architecture can be a good compromise between computing time and quality of reconstructed images at the output system. The principle of all optically architecture implementing the DCT is based on the use of similarity between the Fourier transform (FT) and the DCT and then the use of a converging lens to achieve an optical FT. To achieve this architecture, we begin by duplicating the target image (image to compress) four times in the input plane using a special way [3]. Then, we perform the FT of the obtained plane using a converging lens. After that, we multiply the obtained spectrum with a series of well-defined filters to have the DCT result in the spectral domain. Finally we apply a low pass filter to select only a part of the spectrum (were c denotes the DCT needed size to reconstruct the image, N is the size of the target image). It can be observed in Fig. 9 the different results obtained with different compression ratio defined by: $Cr = 100\left(1 - \frac{c \times c}{N \times N}\right)$. We can easily see the poor quality of the reconstructed image. This is mainly due to the use of different optical components.

5.1 Image Compression Principle and Simulation

A fixed point Matlab Simulink model has been established to validate the proposed method. This step is very important to validate the algorithm structure before the material implementation. This model is very useful for debugging intermediate signals of the hardware description language (HDL).

According to the model of Fig. 8, each image is parallelized into 8-point blocks. This operation can be done by a serial to parallel block composed by 8 flip-flops.

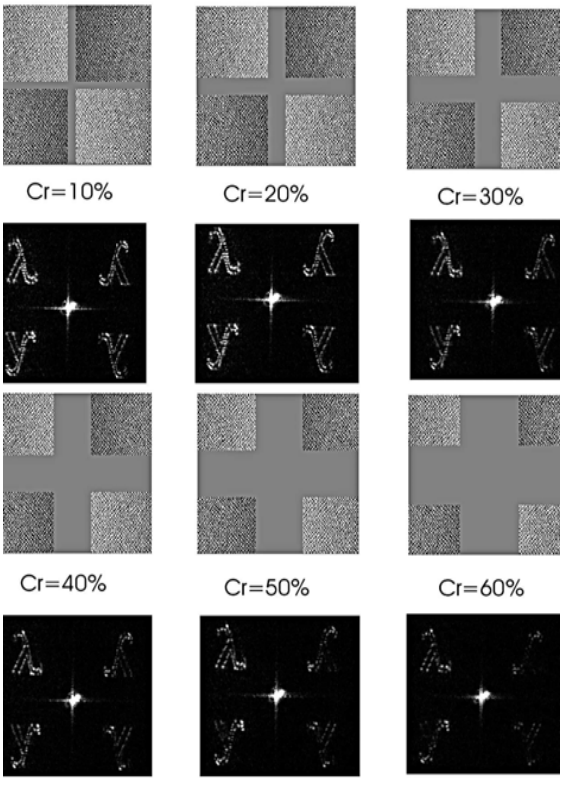


Fig. 9. Optical reconstruction using various compression ratios [3]

After that, the DCT coefficient calculation is performed jointly with a specific quantization process. The basic idea of the quantization consists in considering the DCT low frequencies more than the DCT high frequencies.

The first DCT coefficient can be implemented using an addition tree of 8 input pixels. This coefficient is always considered in all quantization modes without truncation. This means that the word-length of the DC coefficient is equal to 11 bits. A second quantization mode may be defined by considering the first $X\,(1)$ and the second DCT coefficients$X\,(2)$. The associated word-length to $X\,(1)$ and $X\,(2)$ are respectively 11 and 7 bits. Therefore, we can compute two DCT outputs among 8 and we obtain an acceptable compressed image quality. For the decompression process, the 8-point IDCT takes X(1) and X(2) followed by 6 zeros to obtain a sequence of 8 inputs.

Six quantization mode are defined with the same manner by considering higher frequencies using 7 bits for word-length. It can be found that the minimum compression rate is obtained when we take all DCT outputs (from $X\,(1)$ to $X\,(8)$) and the maximum is reached when we take only the first DCT output as low frequency output.

To validate the compression model a 256x256 Lena gray scale image is used, Fig. 10a. Each input block is composed by 8 pixels encoded using 8 bits for each pixel. On the other hand, as stated before, the first DCT coefficient, $X(1)$, is encoded without loss using 11 bits. $X(2)$ to $X(8)$ outputs are encoded using 7 bits. The compression ratio (defined before by Cr depends on the considered number of DCT outputs and may be expressed by the following equation:

$$Cr_i = \left(1 - \frac{(11 + 7 * i)}{8 * 8}\right) * 100 \tag{20}$$

where i is an integer $\in [0, 7]$. According to the last equation, i equal to zero means that we take only the first DCT output and for i equal to 7, all DCT outputs are considered. Finally, PSNR is evaluated and results are summarized in Fig. 10. It can be found that the PSNR decreases with a low values of compression ratio Cr_i. This observation is verified in Fig. 10b to Fig. 10i. According to these images, a compression ratio of 6.25% is equal to $(1 - \frac{(11+7*7)}{64}) * 100$ and means that we take all DCT coefficients (8 points). The PSNR associated to this compression ratio is about 38.2 dB, Fig. 10a. Conversly, a compression ratio of 82.81% is equal to $(1 - \frac{11}{64}) * 100$ and means that we take only one DCT outputs $X(1)$. The associated PSNR to this compression ratio is about 18 dB, Fig. 10i.

5.2 Design Considerations

We use the standard language VHDL for coding which gives the choice of implementing target devices (FPGA family, CPLD, ASIC) at the end of the implementation flow. It means that the image compression model reported here is synthesized and may be implemented on arbitrary technologies.

Moreover, some design considerations are taken into account. In fact, as in [26], we use registered adder and subtractor to perform the high speed implementation. The critical path is minimized by insertion of pipeline registers and is equal to the propagation delay of an adder or a subtractor. It should be outlined that the use of registered outputs comes at no extra cost of an FPGA if an unused D flip-flop is available at the output of each logic cell. For example, to realize an addition of two vectors, the bitstream of the addition adapts the structure of the FPGA to connect slices to each other. These slices are composed of two 4-input LUT and two D flip-flop (four 6-input LUT and 4 D flip-flop for recent Xilinx target). Consequently, for used slices all D flip-flop are free. The registered operator can use these flip-flop to reduce the critical path.

With these considerations, we can confirm that the proposed model is able to be implemented in all digital targets but there are optimized for FPGA devices since they take into account the FPGA structure.

(a) Original

(b) (17.18%, 38.2 dB)

(c) (28.12%, 35.02 dB)

(d) (39.06%, 33.15 dB)

(e) (50%, 31.42 dB)

(f) (60.93%, 28.42 dB)

(g) (71.87%, 24.22 dB)

(h) (82.81%, 22.65 dB)

(i) (93.75%, 18.31 dB)

Fig. 10. Simulation of image compression using the VHDL model

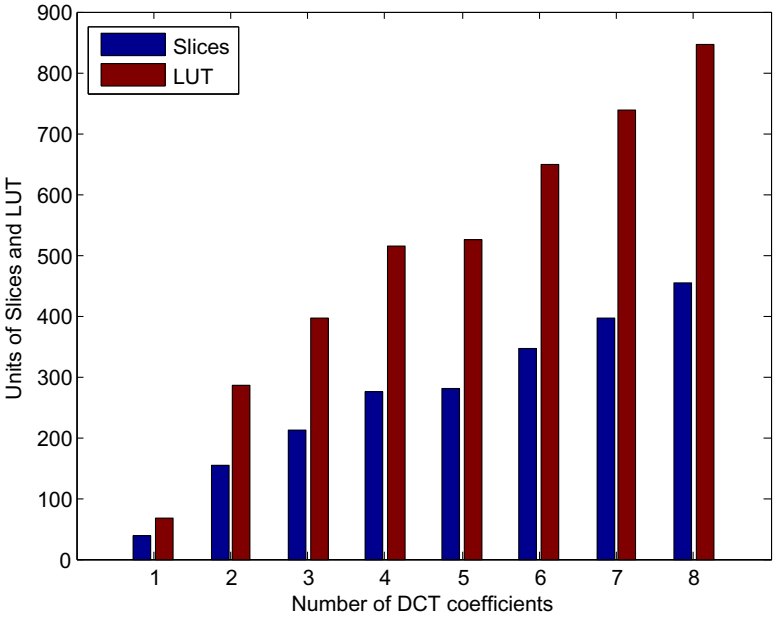


Fig. 11. Slice and LUT resources versus selected number of DCT coefficients

5.3 Synthesis Results

In this section, VHDL programs of different DCT architectures are synthesized using ISE software of Xilinx and the Spartan3E XC3S500 target. In order to illustrate the differences in hardware consumption, the FPGA synthesis results are presented in Table 7. The second column presents required logic and arithmetic elements for the implementation of DCT based on Xilinx embedded multipliers. Column 3 substitutes embedded multipliers MULT18x18 by 4-input LUTs: the total number of required LUTs reaches 1424. Columns 4 and 5 detail the required resources for DCT implementation using respectively CSD encoding and CSD with the proposed CSE technique. Thanks to element sharing, required resources decrease from column 4 to column 5.

Furthermore, the use of the proposed technique involves less computation and consequently high maximum operating frequency. This throughput allows easily the processing of more than 30 frames per second. Finally, note that an old FPGA (Spartan 3E) is used in this work. The offered throughput exceeds 300 MS/s with Virtex 6 target.

Another study related to the joint optimization between DCT computation and quantization process is mentioned in Fig. 11. It can be observed that the number of required LUTs and Slices grows exponentially with the number of selected DCT coefficients. 47 LUTs and 68 Slices are required for the first quantization mode while 450 LUT and 844 Slices are needed to perform the last quantization mode. There is only one case where the required resources present

Table 7. FPGA resources estimation of different DCT design

	Embedded Mult	LUT Mult	CSD	Proposed CSD CSE
Slices	356	-	581	510
Flip-Flop	512	512	454	516
4 input LUTs	404	1424	1012	900
MULT18x18	11	0	0	0
Maximum Frequency (MHz)	119.748	-	118.032	137.678

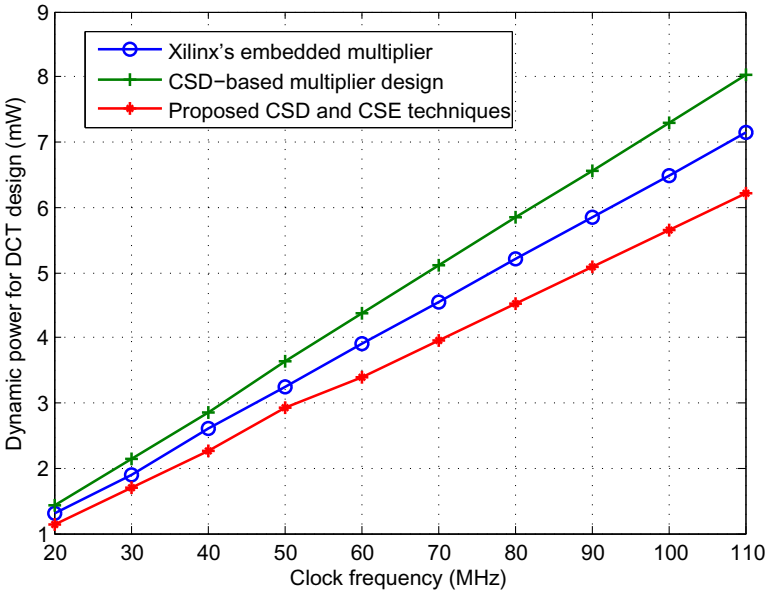


Fig. 12. Dynamic power consumption estimation per sample with 1.6 V design

a slight increase. This case corresponds with the selection of 4 or 5 DCT coefficients. In fact, according to Loeffler algorithm, $X(5)$ requires only one more subtractor compared to the $X(4)$ calculation.

Hence, the image quality can be handled with the required silicon area. Indeed, for available FPGA or ASIC resources a quantization mode can be chosen which involves a certain quality of the reconstructed images. The image quality criteria which is the PSNR and the utilization of FPGA resources are shown in Fig. 10 and Fig. 11 respectively.

5.4 Power Analysis

To have an idea of the estimated power consumption, comparison of the dynamic power between three DCT architectures is made. We used the XPower tool of Xilinx to estimate the dynamic power consumption. It can be found in Fig. 12 and for clock frequency of 110 MHz, the share subexpressions reduces the dynamic power by 22% compared to the CSD-based DCT architecture and by 9% compared to the Xilinx's embedded multipliers-based DCT design . Furthermore, these values are compared favorably with other works as for example 11.5 mW at 2V design and 40 MHz in [36] or 3.94 mw at 1.6 V design and 40 MHz in [27].

6 Conclusion

We have combined three optimization techniques for specific image compression based on the 1D-DCT algorithm. The first one is about the granularity analysis where the 2R4C granularity has been introduced to perform higher image quality. The second optimization concerns the constant coefficient multiplier. The theoretical formulas of DCT coefficients have been derived by applying the CSD encoding and the sharing of common subexpressions. It has been shown that the CSD-based design implies an area and energy economies. Finally, these techniques have been used jointly with a simple and efficient quantization process. Hence, the image quality of compressed image may be handled with the silicon area and power consumptions.

References

1. Alfalou, A., Brosseau, C.: Optical image compression and encryption methods. Adv. Opt. Photon 1, 589–636 (2009)
2. Alfalou, A., Brosseau, C.: Exploiting root-mean-square time-frequency structure for multiple-image optical compression and encryption. Opt. Lett. 35, 1914–1916 (2010)
3. Alkholidi, A., Alfalou, A., Hamam, H.: A new approach for optical colored image compression using the JPEG standards. Journal of Signal Processing 87(4), 569–583 (2007)
4. Jridi, M., AlFalou, A.: A VLSI implementation of a new simultaneous images compression and encryption method. In: IEEE Int. Conf. Imaging Systems and Techniques (IST), pp. 75–79 (July 2010)
5. Yang, E., Wang, L.: Joint Optimization of Run-Length Coding, Huffman Coding, and Quantization Table With Complete Baseline JPEG Decoder Compatibility. IEEE Trans. Image Process. 18(1), 63–74 (2009)
6. Video coding for low bit rate communication (ITU-T Rec. H.263) (February 1998)
7. ISO/IEC DIS 10 918-1, Coding of audio visual objects: part 2. visual, ISO/IEC 14496-2 (MPEG-4 Part2) (January 1999)
8. Chen, W.A., Harrison, C., Fralick, S.C.: A fast computational algorithm for the discrete cosine transform. IEEE Trans. Commun. COM-25, 1004–1011 (1977)
9. Lee, B.: A new algorithm to compute the discrete cosine transform. IEEE Trans. Acoust. Speech Signal Process. ASSP-32, 1243–1245 (1984)

10. Vitterli, M., Nussbaumer, H.: Simple FFT and DCT algorithms with reduced number of operation. Signal Process 6, 264–278 (1984)
11. Suehiro, N., Hatori, M.: Fast algorithms for DFT and other sinusoidal transforms. IEEE Trans. Acoust. Speech Signal Process. ASSP-34, 642–664 (1986)
12. Hou, H.: A fast recursive algorithm for computing the discrete cosine transform. IEEE Trans. Acoust. Speech Signal Process ASSP-35, 1455–1461 (1987)
13. Loeffler, C., Lightenberg, A., Moschytz, G.S.: Practical fast 1-D DCT algorithm with 11 multiplications. In: Proc. ICAPSS, pp. 988–991 (May 1989)
14. Duhamel, P., H'mida, H.: New 2^n DCT algorithm suitable for VLSI implementation. In: IEEE ICAPSS, pp. 1805–1808 (November 1987)
15. Slawecki, D., Li, W.: DCT/IDCT processor design for high data rate image coding. IEEE Trans. Circuits Syst. Video Technol. 2(2), 135–146 (1992)
16. White, S.A.: Application of distributed arithmetic to digital signal processing: a tutorial review. IEEE ASSP Magazine, 4–19 (July 1989)
17. Madisetti, A., Willson, A.N.: A 100 MHz 2-D 8x8 DCT/IDCT processor for HDTV applications. IEEE Trans. Circuit Syst. Video Technol. 5(2), 158–165 (1995)
18. Yu, S., Swartzlander, E.E.: DCT implementation with distributed arithmetic. IEEE Trans. Computers 50(9), 985–991 (2001)
19. Kim, D.W., Kwon, T.W., Seo, J.M., Yu, J.K., Lee, S.K., Suk, J.H., Choi, J.R.: A compatible DCT/IDCT architecture using hardwired distributed arithmetic. In: IEEE Int. Symp. Circuit Syst. (ISCAS 2001), vol. 2, pp. 457–460 (May 2001)
20. Shams, A., Pan, W., Chidanandan, A., Bayoumi, M.: A Low Power High Performance Distributed DCT Architecture. In: IEEE Computer Society Annu. Symp. VLSI (ISVLSI 2002), pp. 21–27 (2002)
21. Meher, P.K.: Unified Systolic-Like Architecture for DCT and DST Using Distributed Arithmetic. IEEE Trans. Circuits Sys I: Regular Papers. 53(12), 2656–2663 (2006)
22. Alam, M., Badawy, W., Julien, G.: A new Time distributed architecture for MPEG-4 hardware reference model. IEEE Trans. Circuit Syst. Video Technol. 15(5), 726–730 (2005)
23. Meher, P.K.: LUT Optimization for Memory-Based Computation. IEEE Trans. Circuits Sys-II, 285–289 (April 2010)
24. Yu, S., Swartzlander, E.E.: A scaled DCT architecture with the CORDIC algorithm. IEEE Trans. Signal Process. 50(1), 160–167 (2002)
25. Sun, C.C., Ruan, S.J., Heyne, B., Goetze, J.: Low-power and high quality Cordic-based Loeffler DCT for signal processing. Circuits, Devices. Syst. 1(6), 453–461 (2007)
26. Jridi, M., AlFalou, A.: Direct digital frequency synthesizer with CORDIC algorithm and Taylor series approximation for digital receivers. European Journal of Scientific Research 30(4), 542–553 (2009)
27. Xanthopoulos, T., Chandrakasan, A.P.: A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization. IEEE Jour. Solid-State Circuits 35(5), 740–750 (2000)
28. Huang, J., Lee, J.: A Self-Reconfigurable Platform for Scalable DCT Computation Using Compressed Partial Bitstreams and BlockRAM Prefetching. IEEE Trans. Circuit Syst. Video Technol. 19(11), 1623–1632 (2009)
29. Huang, J., Lee, J.: Efficient VLSI architecture for video transcoding. IEEE Trans. Consumer Electron. 55(3), 1462–1470 (2009)

30. Hsu, C.L., Cheng, D.H.: Reduction of discrete cosine transform/ quantization/inverse quantization/inverse discrete cosine transform computational complexity in H.264 video encoding by using an efficient prediction algorithm. IET. Image Process. 3(4), 177–187 (2009)
31. Jridi, M., AlFalou, A.: A low-power, high-speed DCT architecture for image compression: Principle and implementation. In: VLSI System on Chip Conference (VLSI-SoC), pp. 304–309 (September 2010)
32. Avizienis, A.: Signed-Digit Number Representations for Fast Parallel Arithmetic. IRE Transaction on Electron. Computer EC-10, 389–400 (1961)
33. Seegal, R.: The canonical signed digit code structure for FIR filters. IEEE Trans. Acoustics Speech Signal Process. 28(5), 590–592 (1980)
34. Hartley, R.I.: Subexpression sharing in filters using canonic signed digit multipliers. IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing 43(10), 677–688 (1996)
35. Pai, C.Y., Lynch, W.E., Al-Khalili, A.J.: Low-Power data-dependant 8x8 DCT/IDCT for video compression. In: IEE, Proc. Vision Image Signal Process., vol. 150, pp. 245–254 (August 2003)
36. Matsui, M., Hara, H., Seta, K., Uetani, Y., Klim, L.S., Nagamatsu, T., Sakura, T.: 200 MHz video compression macrocelles using low-swing differential logic. In: Proceedings of ISSCC, pp. 76–77 (1994)
37. Kim, B., Ziavras, S.G.: Low-power multiplierless DCT for image/video coders. In: IEEE Int. Symp. on Cons. Electronics (ISCE 2009), pp. 133–136 (May 2009)
38. Song, H.S., Cho, N.I.: DCT-based embedded image compression with a new coefficient sorting method. IEEE Sig. Process. Letters. 16(5), 410–413 (2009)