This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

Approximate DCT Image Compression using Inexact Computing

Haider A.F.Almurib, *Senior IEEE*, T. Nandha Kumar, *Senior IEEE*, and Fabrizio Lombardi, *Fellow IEEE*

***Abstract* – This paper proposes a new framework for digital image processing; it relies on inexact computing to address some of the challenges associated with the discrete cosine transform (DCT) compression. The proposed framework has three levels of processing; the first level uses approximate DCT for image compressing to eliminate all computational intensive floating-point multiplications and executing the DCT processing by integer additions and in some cases logical right/left shifts. The second level further reduces the amount of data (from the first level) that need to be processed by filtering those frequencies that cannot be detected by human senses. Finally, to reduce power consumption and delay, the third level introduces circuit level inexact adders to compute the DCT. For assessment, a set of standardized images are compressed using the proposed three-level framework. Different figures of merits (such as energy consumption, delay, power-signal-to-noise-ratio, average-difference, and absolute-maximum-difference) are compared to existing compression methods; an error analysis is also pursued confirming the simulation results. Results show very good improvements in reduction for energy and delay, while maintaining acceptable accuracy levels for image processing applications.**

Keywords—Approximate Computing, DCT, Inexact Computing, Image Compression,

I. INTRODUCTION[1]

Today's computing system usually process a significant amount of information that is computational and power intensive. Digital Signal Processing (DSP) systems are widely used to process image and video information, often under mobile/wireless environments. These DSP systems use image/video compression methods and algorithms. However, the demands of power and performance remain very stringent. Compression methods are often utilized to alleviate such requirements. Image/video compression methods fall into two general categories: lossless and lossy. The latter category is more hardware efficient but at the expense of quality of the final decompressed images/videos. For image processing, the Joint Photographic Experts Group (JPEG) method is the widely used lossy method while the Moving Picture Experts Group (MPEG) method is the widely used lossy method for video processing. Both standards use the Discrete Cosine Transform (DCT) algorithm as basic processing step.

Many different fast algorithms for DCT [1][2] computation have been developed for image and video applications; however as all these algorithms still need floating point multiplications; they are computationally intensive requiring extensive hardware resources. To address these concerns, coefficients in many algorithms such as [3] can be scaled and approximated by integers such that floating-point multiplications can be replaced by integer multiplications [4][5]. The resulting algorithms are significantly faster than the original versions and, therefore, they are extensively used in practical applications. Consequently, the design of good approximations of the DCT for implementation by narrower bus width and simpler arithmetic operations (such as shift and addition) has received considerable attention over the last few years [6].

An advantageous feature of image/video processing is its highly error-tolerant nature; human senses cannot often perceive degradation in performance, such as quality of visual and audio information. Therefore, imprecise computation can be used in many applications that tolerate some loss of precision and some degree of uncertainty, [7][8], such as for example image/video processing.

The introduction of inaccuracy at circuit level in the DCT computation targets specific figures of merit (such as power dissipation, delay and circuit complexity [9]-[14]) and it is very challenging. This scheme targets low power and process tolerance is based on a logic/gate/transistor level redesign of an exact circuit. A logic synthesis approach [9] has been proposed to design circuits for implementing an inexact version of a given function by considering the so-called error rate (ER) as metric for error tolerance. Reduction in circuit complexity at transistor level of an adder circuit (such as by truncating the circuits at the lowest bit positions) provides a reduction in power dissipation higher than conventional low power design techniques [10]; in addition to the ER, new figures of merit for estimating the error in an inexact adder have been presented in [11].

[10] has presented an approximate mirror adder (AMA) circuit that utilizes cell replacement by reducing adder cell circuit complexity compared to a traditional mirror adder (MA) scheme. Cell replacement usually provides a shorter critical path; it also enables voltage scaling and reduces the switching capacitance. [13] has proposed "approximate" adder cells (AXA) based on XOR and XNOR implementations; node capacitances and power are also reduced in the AXA designs of [13].

As relevant to this manuscript, [14] has designed an adder cell circuit with a lower number of transistors such that a reduction in power dissipation is also accomplished. In [14], three new inexact adder cell designs have been presented; they are very favorable for approximate computing in terms of both electrical and error features. Particularly they requires a significantly reduced number of transistors, a small number of erroneous outputs and incur in a substantial reduction in both

Haider A.F.Almurib and T. Nandha Kumar are with the Faculty of Engineering, The University of Nottingham, Malaysia (e-mail: haider.abbas; nandhakumaar.t@nottingham.edu.my). Fabrizio Lombardi is with the Department of ECE, Northeastern University, Boston, MA 02115, USA (e-mail: lombardi@ece.neu.edu).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

delay and energy dissipation. There is an extensive literature on inexact/approximate computing and related hardware design [9]-[15]; however in these papers, the degree of inexactness/approximation is usually restricted to arithmetic functions and/or the final objective(s) is reducing specific figures of merit (such as power/energy consumption, error and delay) mostly at circuit-level. This type of analysis assumes a flat view of operation because it does not take account that many domains of approximation are possible once a specific application (and related algorithms) is considered. This manuscript considers DCT (as one of the most commonly used processing algorithms in DSP) by expanding the domains of approximation into a more comprehensive framework. Therefore, this paper presents a new framework for approximate DCT image compression; this framework is based on inexact computing and consists of three levels.

- Level 1 consists of a multiplier-less DCT transformation, so involving only additions;
- Level 2 consists of high frequency component (coefficient) filtering;
- Level 3 consists of computation using inexact adders.

Level 1 has been widely studied in the technical literature [16][17][18]; Level 2 is an intuitive technique to reduce the complexity of computation while attaining only a marginal degradation in image compression. Level 3 follows a circuit-level technique by which inexact computation is pursued (albeit new and efficient inexact adder cells are utilized in this manuscript). Therefore, the contribution of this manuscript is found in the combined effects of these three levels. The proposed framework has been extensively analyzed and evaluated. Simulation and error analysis show a remarkable agreement in results for image compression as an application of inexact computing.

Hereafter to avoid confusion the word "approximate" is used only for the DCT algorithms while the word "inexact" is used for circuits and design involving non-exact hardware for computing the DCT.

This paper is organized as follows: Section II presents the review of DCT, while approximate DCT implementation using inexact adders are dealt with in Section III. The proposed framework is presented in Section IV, and its evaluation in Section V. Finally, the conclusion is provided in Section VI.

II. Review of DCT

For manuscript completeness, preliminaries to approximate DCT and a review of relevant topics are presented next.

A. Discrete Cosine Transform (DCT)

To obtain the *i*th and *j*th DCT transformed elements of an image block (represented by a matrix p of size N), the following equation is used:

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(x,y) cos\left[\frac{\pi(2x+1)i}{2N}\right] cos\left[\frac{\pi(2y+1)j}{2N}\right]$$
$$C(u) = \begin{cases} 1/\sqrt{2}, & u = 0 \\ 1, & u > 0 \end{cases} \quad (1)$$

where $p(x, y)$ is the x,yth element of the image. This equation calculates one entry (i,jth) of the transformed image from the pixel values of the original image matrix. For the commonly used 8x8 block for JPEG compression, N is equal to 8 and x and y range from 0 to 7. Therefore $D(i,j)$ is also given by the following equation:

$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{i=0}^{7} \sum_{j=0}^{7} p(x,y) cos\left[\frac{\pi(2x+1)i}{16}\right] cos\left[\frac{\pi(2y+1)j}{16}\right] \quad (2)$$

For matrix calculations, the DCT matrix is obtained from the following equation:

$$T_{DCT}(i,j) = \begin{cases} 1/\sqrt{N}, & i = 0 \\ \sqrt{\frac{2}{N}} cos\left[\frac{\pi(2j+1)i}{2N}\right], & i > 0 \end{cases} \quad (3)$$

So, DCT is computation intensive and may require floating-point operations for processing, unless an approximate algorithm is utilized.

B. Joint Photographic Experts Group (JPEG)

The JPEG processing is first initiated by transforming an image to the frequency domain using the DCT; this separates images into parts of differing frequencies. Then, the quantization is performed such that frequencies of lesser importance are discarded. This reflects the capability of humans to be reasonably good at seeing small differences in brightness over a relatively large area, but they usually cannot distinguish the exact strength of a rapidly varying brightness variation. The compression takes place during this quantization step in which each component in the frequency domain is divided by a constant, and then rounded to the nearest integer. This results in many high frequency components having very small or likely zero values, small values at best. The image is then retrieved during the decompression process that is performed using only the important frequencies that have been retained.

For JPEG processing, the following steps must be executed:

1. An image (in color or grey scales) is first subdivided into blocks of kxk pixels (usually k=8).
2. Then from left to right and top to bottom, the DCT is applied to each block.
3. This generates kxk coefficients (so 64 for k=8) that are then quantized to reduce the magnitudes.
4. The resulting array of compressed blocks represents the compressed image, i.e. the stored or transmitted image.
5. To retrieve the image, the compressed image (array of blocks) is decompressed using Inverse DCT (IDCT).

Many different fast algorithms for DCT [1][2] computation have been developed for image and video applications. The method proposed by Loeffler et al. [19] requires 11 multiplications and 29 additions and it is regarded the most efficient because the theoretical lower bound on the number of multiplications required for the 1-D eight-point DCT has been proven to be 11 [20][21]. To achieve a further reduction in computational complexity, some operations of the DCT can be incorporated into the quantization step. These so-called scaled DCTs can result in significant improvements; for example,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

Arai's method needs only five multiplications and a very small number of additions, 29 to be exact [3].

All of the aforementioned fast algorithms still need floating point multiplications, so they are slow and complex in implementation. To achieve a faster computation, coefficients in many algorithms such as Arai's method [3] can be scaled and approximated by integers such that floating-point multiplications can be replaced by integer multiplications [4][5]. The resulting algorithms are significantly faster than their original versions and, therefore, they have wide practical applications.

The fixed-point multiplications required by these fast algorithms generally need a large width data bus (often 32 bits), so a costly VLSI implementation in both hardware and power consumption. Therefore, the design of good approximations of the DCT to be implemented by a narrower bus width and simpler arithmetic operations (such as shift and addition) is very attractive [6].

In recent years, low-complexity methods for the efficient computation of the 8-point DCT can be found in the technical literature [22]. Approximation techniques include the signed DCT (SDCT) [23], the level 1 approximation by Lengwehasatit-Ortega [24], the Bouguezel-Ahmad-Swamy (BAS) series of algorithms [25–27], and the DCT round-off approximation [28]. However, not all approximation techniques found in the literature are beneficial when implementation is considered; for example, [24] requires additional steps prior to computing the DCT to determine the appropriate approximation. In general, the transformation matrix entries for approximate DCT methods are only {0, ±1/2, ±1, ±2}; so a so-called null multiplicative complexity is possible because the involved operations can be implemented exclusively by means of additions and bit-shift operations [29].

These approximate DCT approaches rely on the SDCT for their derivation, as simply defined by applying the signum function operator to the DCT matrix coefficients, i.e.

$$T_{SDCT}(i,j) = \frac{1}{\sqrt{N}} sign\{T_{DCT}(i,j)\}$$

$$sign\{u\} = \begin{cases} +1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \quad (4)$$

Therefore, aside from the $1/\sqrt{N}$ term, the DCT matrix contains only positive/negative ones and zeros, so implying addition/subtraction or no operation.

By setting some of the coefficients of T_{SDCT} to zero, approximate DCTs are possible; so the 8x8 transform matrices C are obtained by a two-step process: (1) introduce few zeros and 1/2s in the 8x8 SDCT matrix to generate a modified transformation matrix T; (2) introduce a diagonal matrix D so that the resulting transformation matrix $C = DT$ is orthogonal, i.e. $C^{-1} = C' = T'D$. Therefore, only a transformation matrix and its transpose are needed to perform the DCT. The signed DCT [23] required two unique transformations (forward and inverse) as T is not orthogonal.

Consider as an example one of the first approximate transformations given by BAS2008 [25];

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 1/2 & -1/2 & -1 & -1 & -1/2 & 1/2 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1/2 & -1 & 1 & -1/2 & -1/2 & 1 & -1 & 1/2 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

and $D = diag\left(\frac{1}{2\sqrt{2}}, \frac{1}{2}, \frac{1}{\sqrt{5}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2}, \frac{1}{\sqrt{5}}, \frac{1}{\sqrt{2}}\right)$.

As an example, let X be an 8x8 block matrix of an image and F its corresponding block matrix in the transform domain. Then, the forward and inverse transformation operations can be performed using the approximate transform as $F = CXC' = D(TXT')D$ and $X = C'FC = T'(DFD)T$ respectively. The above approximate transform (as well as all other transforms found in the literature) is not multiplication free; however, it can be made so by merging the diagonal matrix D into the quantization/de-quantization matrix because the quantization/de-quantization is applied to the block matrix in the transform domain. In comparison with a conventional DCT transformation and given the exact transformation matrix T, $F = TXT'$ and the reverse is $X = T'F'T$, where F' is the decoded image portion.

Table 1 illustrates the 8-point DCT computation complexity of the approximation DCT methods and a comparison with the original DFT, Cooley–Tukey DFT and the original DCT.

Table 1: Approximate DCT methods applied to image compression; number of operations required to calculate the DCT for a 8x8 block size.

	Method	Additions	Multiplications	Shifts	Total operations
Multipliers	DFT by definition [1]	56[a] (432)	64[b] (192)	0	624
	DFT, Cooley–Tukey [1]	24[a] (58)	2[b] (6)	0	64
	DCT by definition [2]	56	64	0	120
	Arai algorithm [3]	29	5	0	34
Multiplier-less	SDCT [23]	24	0	0	24
	BAS08 [25]	18	0	2	20
	BAS09 [26]	18	0	0	18
	BAS11 [27] with a = 0	16	0	0	16
	BAS11 [27] with a = 1	18	0	0	18
	BAS11 [27] with a = 2	18	0	2	20
	CB11 [28]	22	0	0	22
	BC12 [29]	14	0	0	14
	PEA12 [16]	24	0	6	30
	PEA14 [17]	14	0	0	14

[a] Complex additions. One complex addition represents two real additions [1]. The equivalent complexity in real additions is in parenthesis.

[b] Complex multiplications. One complex multiplication represents three real multiplications and five real additions [1]. The equivalent complexity in real multiplications is in parenthesis.

In addition to DCT matrix manipulation approximation, fast algorithms for the computation of the DCT such as using multiplier-free approximations [18] and pixel level behavioral decision [30] have been proposed. In general, these approximate methods rely on a trade-off between accuracy and low power.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

III. Inexact Addition and Approximate DCT

Arithmetic circuits are well suited to inexact computing; addition has been extensively analyzed in the technical literature and is one of the fundamental arithmetic operations in many applications of inexact computing [31]. A reduction in circuit complexity at transistor level of an adder circuit usually provides a good reduction in power dissipation, often higher than conventional low power design techniques [10]. Inexact adder designs have been evaluated in [12]: inexact operation has been introduced by either replacing the accurate cell of a modular adder design with an approximate cell of lower circuit complexity, or by modifying the generation and propagation of the carry in the addition process.

In [14], three new inexact adder cell designs have been presented (denoted as InXA1, InXA2 and InXA3); these cells have both electrical and error features that are very favorable for approximate computing. These adder cells as shown in Table 2 have the following advantageous features over previous designs [10][13]: (i) a very small number of transistors; (ii) a very small number of erroneous outputs at the two outputs (i.e. Sum and Carry); (iii) smaller switching capacitances (expressed in *Cgn* gate capacitance of minimum size NMOS), thus incurring in a substantial reduction in both delay and energy dissipation (Table 3) (and their product as combined metric).

Table 2. Error features of different inexact adder cells

Approach	No. of transistors	No. of erroneous values		Error rate %		Total node capacitance in *Cgn*
		Sum	Carry	Sum	Carry	
AMA1, [10]	20	2	1	25	12.5	40
AMA2, [10]	14	2	0	25	0	32
AMA3, [10]	11	3	1	37.5	12.5	27
AMA4, [10]	15	3	2	37.5	25	29
AXA, [13]	7	2	0	25	0	14
InXA1, [14]	6	0	2	0	25	20
InXA2, [14]	8	2	0	25	0	14
InXA3, [14]	6	2	0	25	0	12

Metrics such as delay, energy dissipated and EDP (energy delay product) of the inexact cells for both average and worst cases are presented in Table 3. Among the inexact cells, InXA1 has the least average and worst case delays while InXA2 incurs in the least average and worst case power dissipations and least average EDP. The average and worst case delays and energy dissipation of the adder cells have been determined by exhaustive simulation. For each input signal, the delay is measured when the output reaches 90% of the maximum value while the energy dissipated is measured in all transistors during the time when the output reaches 90%. As per these advantages, InXA1 and InXA2 based adders are considered for the DCT application as treated next.

Table 3. Electrical features of different inexact adder cells at 45nm PTM model [32] and using exhaustive inputs

Approach	Average delay (ps)	Average energy dissipated (fJ)	Average EDP (ps.fJ)	Worst delay (ps)	Worst energy dissipated (fJ)	Worst EDP (ps.fJ)
EFA, [33]	174.419	0.926751	161.643	424.647	2.366840	1005.07
AMA1, [10]	16.539	0.513076	8.486	22.147	0.97947	21.69
AMA2, [10]	25.314	0.663119	16.7863	27.64	0.72036	19.91
AMA3, [10]	23.647	0.664915	15.7236	25.64	0.71163	18.25
AMA4, [10]	17.647	0.478031	8.4361	26.64	0.62711	16.71
AXA, [13]	30.841	0.404277	12.4684	35.64	0.892418	31.812
InXA1, [14]	9.647	0.153591	1.4817	13.64	0.209684	2.861
InXA2, [14]	16.814	0.056326	0.9470	43.14	0.129162	5.5730
InXA3, [14]	59.8	0.340974	20.4043	80.64	0.421182	33.9673

Table 4 summarizes the simulation results of implementing approximate DCT methods using InXA1 and InXA2 based adders, the two methods that resulted in the lowest EDPs [14], for different NAB values, where NAB is defined as follows.

Definition: The Number of Approximate Bits (NAB) is defined as the number of bits starting from the LSB that utilize inexact cells.

The data in this table are the results of using the different inexact addition approaches to calculate the DCT transform for one 8x8 image block-size. To obtain the figures of Table 4, the average case savings of a single adder cell with exhaustive inputs from Table 3 were used in conjunction with Table 1 to calculate the delay, energy dissipated and EDP results of all approximate DCT methods. Please note here that to simplify the calculations, the shift operations that are required by BAS08, BAS11(a=2) and PEA12 were not considered. Table 3 also shows the results from implementing truncations using NAB values of 3, 4 and 5 bits. The adders used to obtain the results of Tables 3 all are of 16-bits sizes.

Table 4. Average Metrics for Approximate DCT using InXA1, InXA2 and truncated addition compared with exact computation.

Method	NAB = 3									NAB = 4									NAB = 5									Exact		
	Delay (ns)			Energy (fJ)			EDP (ns.fJ) x 10^3			Delay (ns)			Energy (fJ)			EDP (ns.fJ) x 10^3			Delay (ns)			Energy (fJ)			EDP (ns.fJ) x 10^3			Delay (ns)	Energy (fJ)	EDP (ns.fJ) x 10^3
	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc	InXA1	InXA2	Trunc			
SDCT [23]	55.1	55.6	54.4	300	293	289	16.5	16.3	15.7	51.2	51.9	50.2	282	272	267	14.4	14.1	13.4	47.2	48.1	46.1	263	251	245	12.4	12.1	11.3	67.0	356	23.8
BAS08 [25]	41.3	41.7	40.8	225	220	217	9.3	9.2	8.9	38.4	38.9	37.7	211	204	200	8.1	7.9	7.5	35.4	36.1	34.5	197	189	184	7.0	6.8	6.3	50.2	267	13.4
BAS09 [26]	41.3	41.7	40.8	225	220	217	9.3	9.2	8.9	38.4	38.9	37.7	211	204	200	8.1	7.9	7.5	35.4	36.1	34.5	197	189	184	7.0	6.8	6.3	50.2	267	13.4
BAS11 [27] a = 0	36.7	37.1	36.3	200	195	193	7.4	7.2	7.0	34.1	34.6	33.5	188	182	178	6.4	6.3	6.0	31.5	32.0	30.7	175	168	163	5.5	5.4	5.0	44.7	237	10.6
BAS11 [27] a = 1	41.3	41.7	40.8	225	220	217	9.3	9.2	8.9	38.4	38.9	37.7	211	204	200	8.1	7.9	7.5	35.4	36.1	34.5	197	189	184	7.0	6.8	6.3	50.2	267	13.4
BAS11 [27] a = 2	41.3	41.7	40.8	225	220	217	9.3	9.2	8.9	38.4	38.9	37.7	211	204	200	8.1	7.9	7.5	35.4	36.1	34.5	197	189	184	7.0	6.8	6.3	50.2	267	13.4
CB11 [28]	50.5	51.0	49.9	275	269	265	13.9	13.7	13.2	46.9	47.5	46.1	258	250	245	12.1	11.9	11.3	43.3	44.1	42.2	241	230	224	10.4	10.2	9.5	61.4	326	20.0
BC12 [29]	32.2	32.5	31.7	175	171	169	5.6	5.6	5.4	29.8	30.2	29.3	164	159	156	4.9	4.8	4.6	27.5	28.0	26.9	153	147	143	4.2	4.1	3.8	39.1	208	8.1
PEA12 [16]	55.1	55.6	54.4	300	293	289	16.5	16.3	15.7	51.2	51.9	50.2	282	272	267	14.4	14.1	13.4	47.2	48.1	46.1	263	251	245	12.4	12.1	11.3	67.0	356	23.8
PEA14 [17]	32.2	32.5	31.7	175	171	169	5.6	5.6	5.4	29.8	30.2	29.3	164	159	156	4.9	4.8	4.6	27.5	28.0	26.9	153	147	143	4.2	4.1	3.8	39.1	208	8.1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

The following conclusions can be drawn from the previously presented results.

- The utilization of InXA2 results in a better EDP than InXA1. As shown in subsequent sections, truncation yields a lower EDP but it incurs in low quality results when used for image compression
- As the shift operations were not included in the comparison of the approximate DCT methods, it has been found that BC12 and PEA14 require the least amount time and energy to perform the 8x8 block DCT transformation.
- The algorithm that has the most delay and dissipates the most energy for DCT is PEA12 and SDCT.

IV. Proposed Approximate Framework

This paper presents a new image compression framework that consists of three levels of approximation as follows.

- Level 1 is the multiplier-less DCT transformation,
- Level 2 is the high frequency filtration,
- Level 3 is the inexact computation.

Levels 1 and 3 were explained in previous sections. Although high frequency filtration (Level 2) is not a new concept, it is appropriate to describe it for sake of completeness because it contributes to the proposed framework for reducing its execution time and energy.

Therefore, instead of performing the quantization process on all resulting DCT transformation coefficients, the process is only performed on the set of coefficients for the low frequency components of the transformed block.

A. High frequency filtration

Filtering the high frequencies generates an image that is hardly distinguished by the human eye (as only sensitive to low frequency contents).

This feature can be used to compress an image. As outlined earlier, a DCT transforms the image in the frequency domain such that it is possible to ignore those coefficients that encode the high frequency components (so not sensitive to the human eye) while retain the other coefficients.

Different numbers of retained coefficients are considered when applied to image compression applications; it has been demonstrated that just 0.34% – 24.26% out of 92112 DCT coefficients are sufficient in high speed face recognition applications [34][35]. Examples for 8x8 image blocks are as follows:

- Image compression using a supporting vector machine in which only the first 8–16 coefficients are considered [36],
- An image reconstruction method based on three coefficients only as proposed in [37],
- Evaluation and assessment of various image compression methods employing only 10 coefficients as in [25][26].

B. Approximate DCT implementation

Unlike the implementations of approximate DCT approaches found in Table 1, next all required calculations (addition and subtraction) are implemented at bit level using the corresponding logic functions. The length of all operators is given by 32-bits and implementations are simulated by MATLAB using their Boolean logical functions.

Selected approximate DCT approaches are simulated for the Lena image; the results are plotted in Figure 1 in which the Power Signal to Noise Ratio (PSNR) of all methods is plotted against the number of Retained Coefficients (RC) used in the quantization stage of the compression.

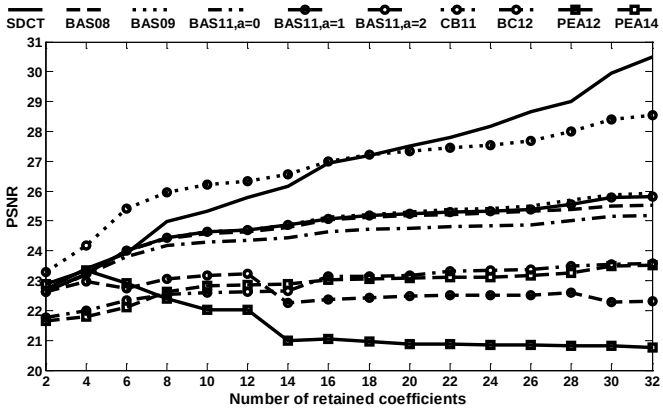

Figure 1 Compression of Lena using Approximate DCT and Bit-level Exact Computing.

The PSNR is calculated from the Mean Square Error (MSE) as follows:

- *Mean Square Error (MSE):*

$$\mathrm{MSE} = \frac{1}{m \times n} \sum_{x=1}^{m} \sum_{y=1}^{n} \left(p_{x,y} - \hat{p}_{x,y}\right)^2 \quad (5)$$

where $p_{j,k}$ is the accurate pixel value at row x and column y of the image, $\hat{p}_{x,y}$ is the approximate value of the same pixel, m and n are the size of the image (rows and columns respectively).

- *Peak Signal to Noise Ratio (PSNR):*

$$\mathrm{PSNR} = 10 \log \frac{(2^n - 1)^2}{\mathrm{MSE}} \quad (6)$$

The results show that, except for the non-orthogonal SDCT method, compression using CB11 generally produces the best outcome in terms of PSNR. Three types of behavior are observed.

- Increasing output quality with an increase of the number of retained coefficients (RC). This occurs for CB11, BAS08, BAS09, BAS11(a=0 and a=1),
- An almost constant PSNR by increasing the RC. This occurs for BC12 and PEA14,
- Degradation in output quality with an increase of RC. This occurs for both BAS11(a=2) and PEA12.

Two additional measures are used for a better insight on the resulting quality, i.e. the Average Difference (AD) and the Maximum Absolute Difference (MD). These metrics are defined as

- *Average Difference (AD):*

$$\mathrm{AD} = \frac{1}{m \times n} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(p_{x,y} - \hat{p}_{x,y}\right) \quad (7)$$

- *Maximum Absolute Difference (MD):*

$$\mathrm{MD} = \max_{m,n} \left\{ \left| p_{x,y} - \hat{p}_{x,y} \right| \right\} \quad (8)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

Figures 2 and 3 show the resulting AD and MD for all methods; the average difference between the uncompressed and inexact-compressed images become smaller as RC increases except for BAS11(a=2) and PEA12 (further confirming the PSNR results in Figure 1). Figure 3 shows that the MD between the uncompressed and inexact-compressed image pixels is reduced as more retained coefficients are used, the exceptions are PEA12 and BAS11 (a=2). This further confirms the previous results.

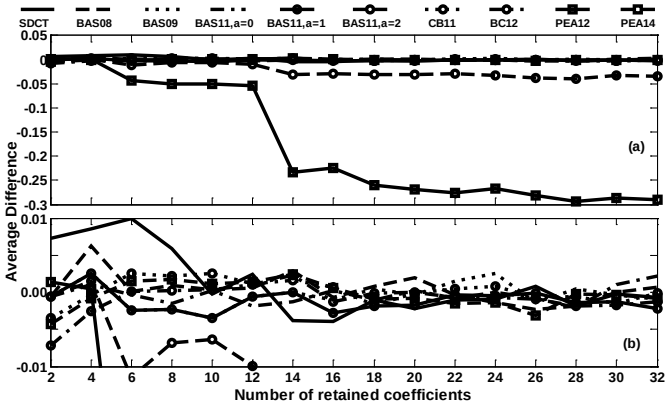

Figure 2 Average Difference (AD) for compression of Lena using Approximate DCT and Bit-level Exact Computing. (a) Full scaled results, and (b) Zoomed out results.

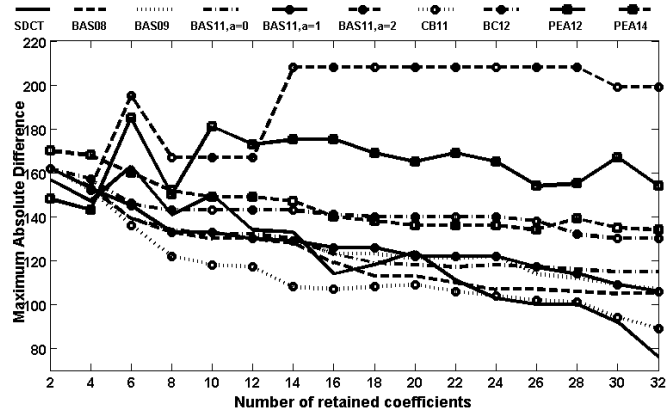

Figure 3 Maximum Absolute Difference (MD) for compression of Lena using Approximate DCT and Bit-level Exact Computing.

Figure 4 depicts the compressed Lena image using the most accurate CB11 method for three RC values, i.e. 4, 10 and 16 retained coefficients. This figure also shows for comparison purpose the exact DCT compression results with RC = 16.

Figure 4 Compression of Lena using CB11 Approximate DCT method and Bit-level Exact Computing; (a) RC = 4, (b) RC = 10, (c) RC = 16, and (d) Exact DCT compression with RC = 16.

C. *Approximate DCT using inexact computing*

Consider next the approximate DCT compression of Lena using inexact adders; as previously, the value of the NAB is increased from 3 to 5. The PSNR results are shown in Figure 6 versus RC; the PSNR of the compressed images (a measure of quality) is plotted by executing all approximate DCT methods using only one inexact adder (for example the top row uses AMA1 as the inexact adder). Each column plots the quality of the compressed images by executing all approximate DCT methods with only inexact adders (for a NAB value). For example the left most column are for NAB=3. As expected, the PSNR deteriorates as the NAB increases (an acceptable level of PSNR is reached at a NAB value of 4).

D. *Truncation*

Truncation is one of the possible inexact computing techniques that may be utilized; the results of using truncation are shown in Figure 5. The use of inexact adders results in more accurate results (truncation is performed at values of 3 and 4 bits).

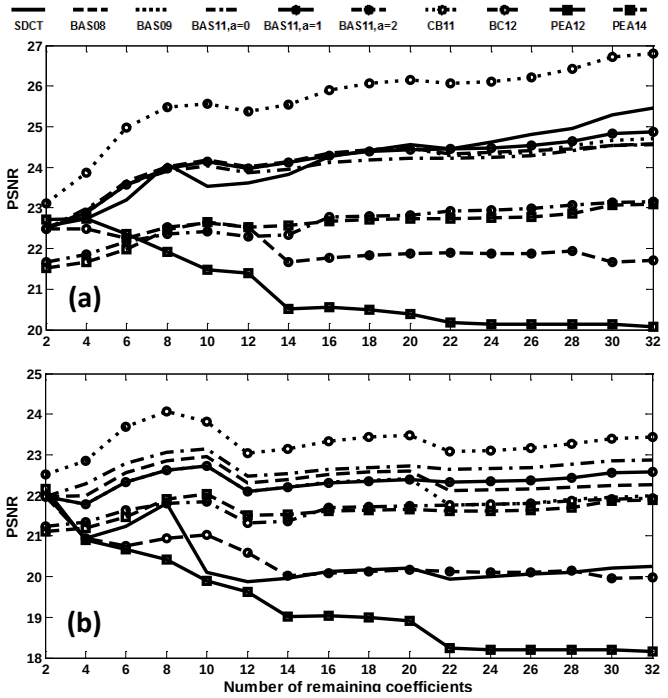

Figure 5 Approximate DCT compression of Lena using Truncation; (a) 3 LSB bits truncation and (b) 4 LSB bits truncation.

E. *Images*

In previous sections, only Lena was utilized as benchmark image. In this section three other images are used to verify the validity of the results. Figures 7 and 8 show the results of compressing Cameraman, Boat and Peppers using NAB values of 3 and 4. The results show the same trends as for Lena, thus confirming the results previously presented.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

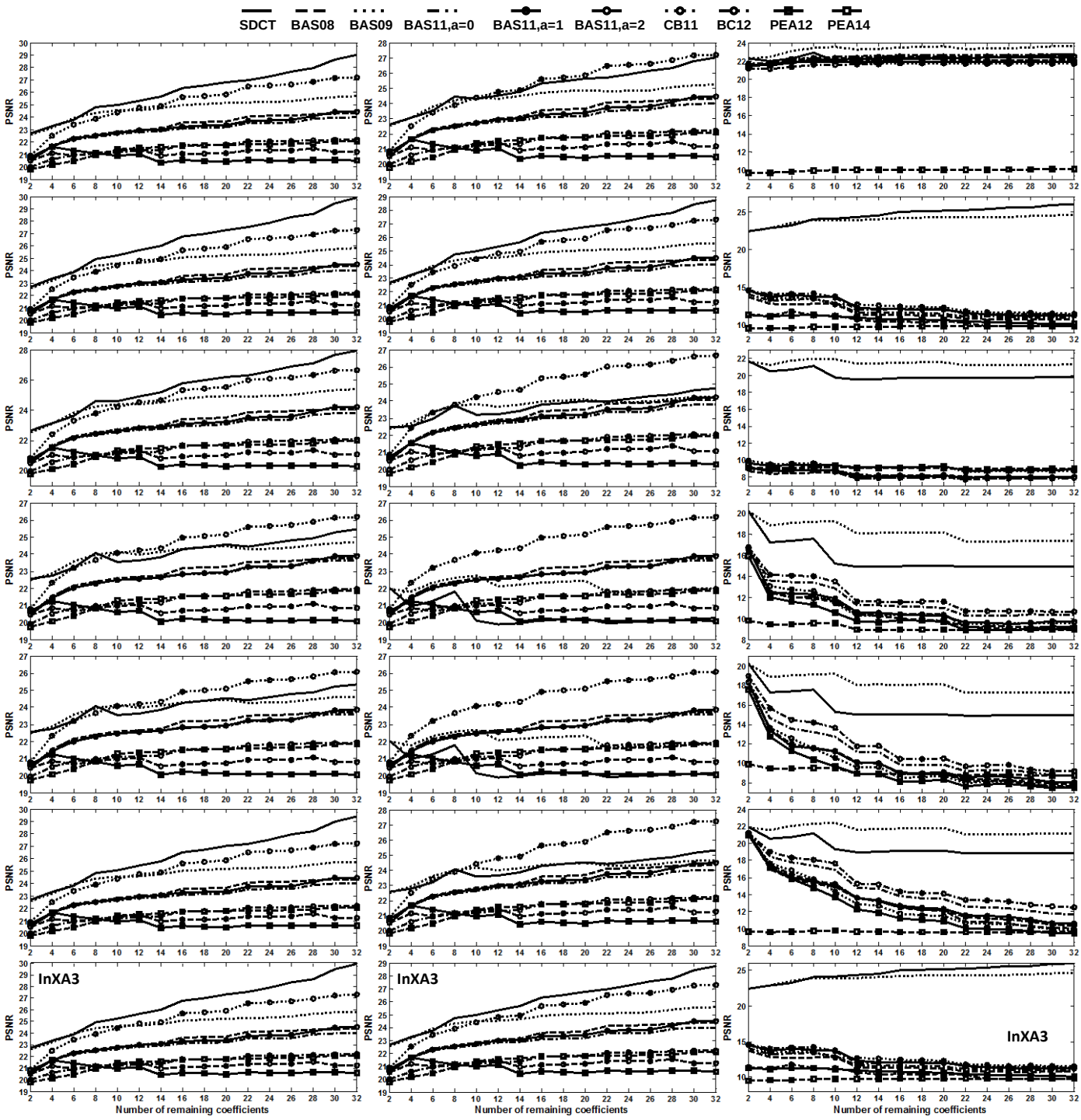

Figure 6 Approximate DCT compression of Lena using inexact adders with different NAB values; (a) NAB=3, (b) NAB=4, and (c) NAB=5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

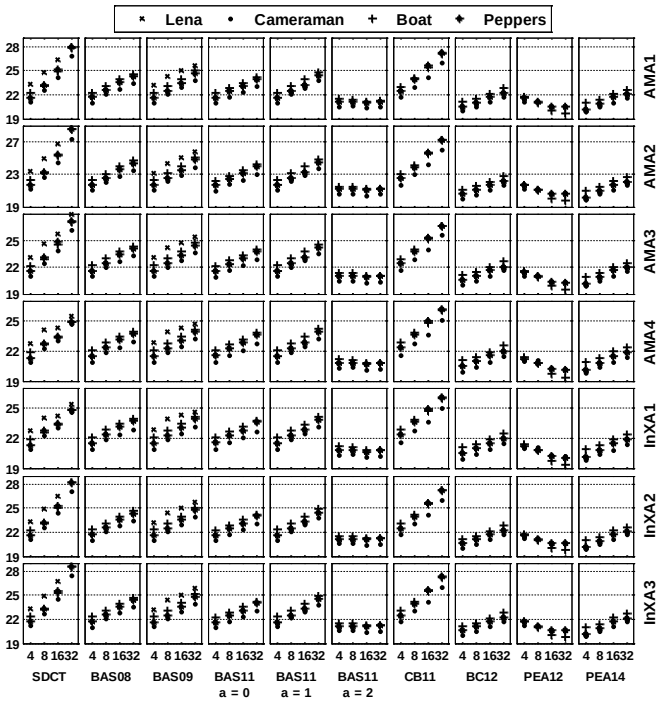

Figure 7 Approximate DCT compression of Lena, Cameraman, Boat and Peppers using Approximate Adders with NAB = 3.
Horizontal axis = retained coefficients. Vertical axis = PSNR.

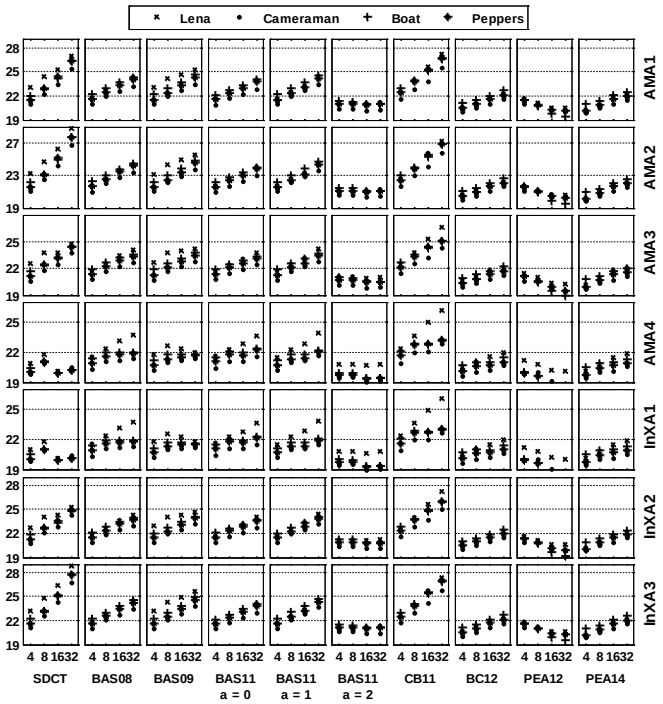

Figure 8 Approximate DCT compression of Lena, Cameraman, Boat and Peppers using Approximate Adders with NAB = 4.
Horizontal axis = retained coefficients. Vertical axis = PSNR.

V. EVALUATION

Next, a discussion will be presented with emphasis on inexact addition and approximate DCT methods for image compression. In such discussion, InXA2 inexact adders are used because it has been shown that they have the best performance amongst all inexact adders (shown in Section III and in [14]). Initially, the number of bits in an adder and the value of the NAB are considered; subsequently, image compression results are discussed and all approximate DCT methods using inexact additions are ranked.

For a given n-bits inexact adder, higher the NAB value, lower are the delay and the energy required to perform an addition. This is illustrated in Table 5 in which the same ratio of n/NAB is considered (for different values of n). The reduction is given in percentage as

$$Reduction = (V_{inxt} - V_{ext})/V_{ext} * 100\% \quad (9)$$

where V_{inxt} is the variable (delay or energy) value for perform an n-bits addition using an inexact adder with a given NAB, and V_{ext} is the same variable value when executed on an exact adder.

The value of V_{inxt} is given by:

$$V_{inxt} = P_{inxt}\ NAB\ +\ (n - NAB)\ P_{ext} \quad (10)$$

where P_{inxt} is the average parameter (delay or energy) value of a single cell inexact adder while P_{ext} in the corresponding parameter value for the exact single cell adder.

Table 5 includes as an example the reduction in delay under various n/NAB ratios and at different values of n.

Table 5: Reduction in time and energy when using inexact adders.

n/NAB ratio	**4**	**2**	**1**
Reduction, (%)	24.85	49.70	99.40
Delay example			
Delay (ns), n=4, $V_{ext} = 11.17\ ns$	8.39	5.62	0.07
Delay (ns), n=8, $V_{ext} = 22.33\ ns$	16.78	11.23	0.13
Delay (ns), n=16, $V_{ext} = 44.67\ ns$	33.57	22.47	0.27
Delay (ns), n=32, $V_{ext} = 89.33\ ns$	67.13	44.94	0.54

With InXA2 and a NAB value of 1, the addition results in no error and therefore the PSNR is not degraded, so the values of reduction in delay/energy are at the lowest (Table 6 shows the reductions at NAB=1 for different values of n).

Table 6: Reduction in delay time and energy when using inexact adders.

n	**4**	**8**	**16**	**32**
Reduction, (%)	24.85	12.42	6.21	3.11

The execution time of an approximate DCT method is affected by the delay time of an addition and the number of additions in the method itself (as reported in Table 1).

If the application of image compression using approximate DCTs is sought, then n should be considered very carefully. Since the depth of considered images is 8 bits, n should be chosen by considering both the largest number of additions required to perform any of the approximate DCT methods (i.e. 24 in Table 1) as well as the depth of the considered images (i.e. 8); so in the worst case, n should be 8+24, or 32 bits. For image quality, previous sections have shown that a NAB value of 4 generates a reasonable balance between time/energy reductions and the resulting quality of the compressed image (as measured by the PSNR).

Table 7 summarizes the results of using an InXA2 based inexact adder to compress 256x256 images for the approximate DCT JPEG compression methods discussed in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2731770, IEEE Transactions on Computers

this work. The four images used in this manuscript are Lena, Cameraman, Boat and Peppers; again, the adder size is n=32 and the NAB value is 4.

The first column in Table 7 shows the ranking of the approximate methods with respect to the reductions of execution time/energy for the different approximate DCT methods; in this case, the reduction depends also on the number of additions a specific approximate DCT method requires to calculate the DCT matrix of an 8x8 size block. The next three columns show the average execution time reduction (which is related to the delay reduction as reported previously in Table 5 for the InXA2-based inexact adder), the average energy reduction (as per (9)) and the average PSNR (as averaged for the four considered images). The last column ranks the approximate DCT methods according to the generated quality of the compressed images (i.e. by the average PSNRs in the fourth column).

Table 7 shows that on average the BC12 and PEA14 methods require the least execution time and energy to compress an image compared to compressing an image using an exact adder. However, with regards to the quality of the picture, BC12 yields poor quality and it is ranked 9 (Table7). As for the best PSNR, the approximate DCT method CB11 produces the highest value; however if both the rankings in the first and fifth columns are equally considered, then BAS09 is the best approximate method.

Table 7: Ranking and average metrics (reductions in execution time/energy and PSNR) of approximate DCT methods for image compression.

Method	Reduction Rank	Average Time reduction (%)	Average Energy reduction (%)	Average PSNR	Quality Rank
SDCT [23]	9	22.54	23.60	22.76	3
BAS08 [25]	4	22.51	23.60	22.64	5
BAS09 [26]	4	22.51	23.60	22.97	2
BAS11 [27] $a = 0$	3	22.60	23.21	22.46	7
BAS11 [27] $a = 1$	4	22.51	23.60	22.61	6
BAS11 [27] $a = 2$	4	22.51	23.60	21.08	10
CB11 [28]	8	22.64	23.31	23.93	1
BC12 [29]	1	22.76	23.56	21.10	9
PEA12 [16]	9	22.54	23.60	20.68	4
PEA14 [17]	1	22.76	23.56	21.28	8

As for the visual impact of errors, Figure 9 illustrates the compression of Lena using all considered approximate DCT methods with a RC value of 10 (as the recommended RC value by [17][25][26]) and a InXA2-based inexact adder; the approximate DCT method CB11 confirms the ranking in the fifth column of Table 7. The error analysis is provided in the supplemental material.

VI. Conclusion

This paper has presented a new approach for compressing images by approximate compression using the Discrete Cosine Transform (DCT) algorithm. The proposed approach consists of a 3-level framework by which initially a multiplier-less DCT transformation (so involving only additions and shift operations) is executed; this level is followed by a high frequency component (coefficient) filtering and computation using inexact adders. It has been shown that using 8x8 image blocks each level contributes to an approximation in the compression process, while still generating at the end a very high quality image. This manuscript has confirmed that the combined effects of these three levels are well understood; simulation and error analysis have shown a remarkable agreement in results for image compression as an application of inexact computing.

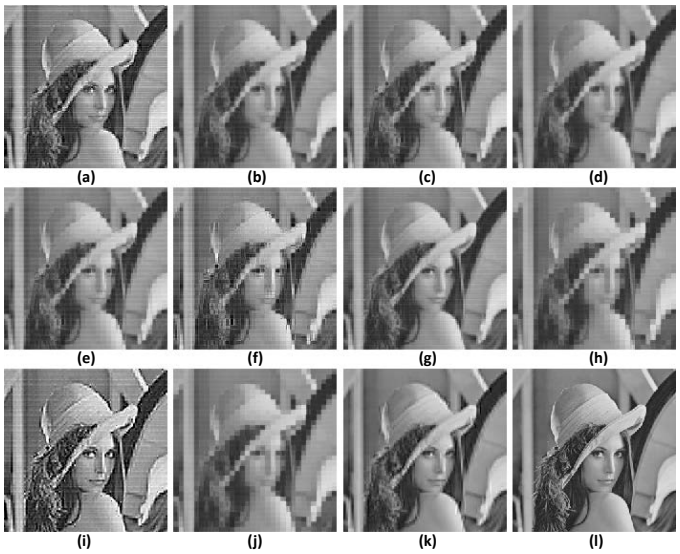

Figure 9 Compression of Lena under considered approximate DCT methods and using an InXA2-based inexact adder with RC = 10; (a) SDCT [23], (b) BAS08 [25] , (c) BAS09 [26] , (d) BAS11 [27]a = 0, (e) BAS11 [27] a = 1, (f) BAS11 [27]a = 2, (g) CB11 [28] , (h) BC12 [29], (i) PEA12 [16], (j) PEA14 [17], (k) Exact DCT compression with RC=10, and (l) Original uncompressed image.

As the proposed framework has been proved to be effective for a DCT method combining approximation at all three proposed levels, the following specific findings have been found and confirmed in this manuscript by simulation and error analysis.

- Among all approximate DCT methods, CB11 produces the best quality compression (highest PSNR values) when using exact 16 bits adders (Figure 1). Other image manipulation quality measures (AD and MD) confirmed the PSNR results. (Figures 2 and 3). Methods BAS08, BAS11 with a=0 and BAS11 with a=1 are the next best methods.
- Among all inexact adders discussed [14], it has been found that InXA2 performs the best.
- When inexact adders are utilized to implement approximate DCT JPEG compression, non-truncation based methods produces better results than the corresponding truncation schemes, especially when considering higher NABs. (Figures 5 and 6).
- The results for the DCT computed by using inexact adders are consistent when different images were used. (Figures 7 and 8). In general acceptable compression can be obtained with NAB values up to 4. Then it has been

shown that the quality of the results decreases substantially when larger NAB values are used.

- On average using 4 image benchmarks, the BC12 and PEA14 methods take the least execution time and energy to compress an image compared to compressing an image using an exact adder. As for the best PSNR as metric of image quality, the approximate DCT method CB11 produces the highest value; however if both reductions in execution time and energy are considered, then BAS09 is the best approximate DCT method.

REFERENCES

[1] R. E. Blahut, Fast Algorithms for Digital Signal Processing. Reading, MA: Addison-Wesley, 1985

[2] V. Britanak, P. Yip and K. R. Rao, Discrete Cosine and Sine Transforms, New York: Academic, 2007

[3] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," Trans. IEICE, vol. E-71, no. 11, pp. 1095–1097, Nov. 1988.

[4] Yun and S. Lee, "On the fixed-point-error analysis of several fast DCT algorithms," IEEE Trans. Circuits Syst. Vid. Technol., vol. 3, pp. 27–41, Feb. 1993.

[5] C. Hsu and J. Yao, "Comparative performance of fast cosine trans- form with fixed-point roundoff error analysis," IEEE Trans. Signal Processing, vol. 42, pp. 1256–1259, May 1994.

[6] Jie Liang; Tran, T.D., "Fast multiplierless approximations of the DCT with the lifting scheme," IEEE Transactions on Signal Processing, vol.49, no.12, pp.3032-3044, Dec 2001.

[7] Y. Dote and S.J. Ovaska, "Industrial Applications of Soft Computing: A Review," Proc. IEEE, vol. 89, no. 9, pp. 1243-1265, Sept. 2001.

[8] K. V. Palem, "Energy Aware Computing through Probabilistic Switching: A Study of Limits," IEEE Trans. Computers, vol. 54, no. 9, pp. 1123-1137, Sept. 2005.

[9] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in Proc. Design, Automat. Test Europe, 2010, pp. 957–960.

[10] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32,no. 1, pp. 124–137, Jan 2013.

[11] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Transactions on Computers, vol. 62, no. 9, pp. 1760–1771, 2013.

[12] H. Jiang, J. Han and F. Lombardi, "A Comparative Review and Evaluation of Approximate Adders," Proc. ACM/IEEE Great Lakes Symposium on VLSI, pp. 343-348, Pittsburgh, May 2015.

[13] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based Adders for Inexact Computing," Proceedings of the IEEE International Conference on Nanotechnology, pp. 690-693, Beijing, August 2013.

[14] Haider A.F.Almurib, T. Nandha Kumar and F. Lombardi," Inexact Designs for Approximate Low Power Addition by Cell Replacement" Proc. IEEE DATE, pp. 660-665, Dresden, March 2016.

[15] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in Proc. Design, Automat. Test Europe, 2007, pp. 1–6.

[16] U. S. Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, and N. Rajapaksha, "Multiplier-free DCT approximations for RF multi-beam digital aperture-array space imaging and directional sensing," Meas. Sci. Technol., vol. 23, no. 11, pp. 1–15, Nov. 2012.

[17] U. S. Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Edirisuriya, "Improved 8-Point Approximate DCT for Image and Video Compression Requiring Only 14 Additions," IEEE Transactions on Circuits and Systems I: Regular Papers, vol.61, no.6, pp.1727-1740, 2014.

[18] Merhav, N.; Vasudev, B., "A multiplication-free approximate algorithm for the inverse discrete cosine transform," in Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on , vol.2, no., pp.759-763 vol.2, 24-28 Oct. 1999.

[19] C. Loeffler, A. Lightenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," Proc. IEEE ICASSP, vol. 2, pp. 988–991, Feb. 1989.

[20] P. Duhamel and H. H'Mida, "New S DCT algorithms suitable for VLSI implementation," in Proc. ICASSP, 1987, pp. 1805–1808.

[21] E. Feig and S. Winograd, "On the multiplicative complexity of discrete cosine transform," IEEE Trans. Inform. Theory, vol. 38, pp. 1387–1391, July 1992.

[22] Lecuire, V.; Makkaoui, L.; Moureaux, J.-M., "Fast zonal DCT for energy conservation in wireless image sensor networks," in Electronics Letters , vol.48, no.2, pp.125-127, January 19 2012.

[23] T.I. Haweel, "A new square wave transform based on the DCT", Signal Processing, vol. 82, pp. 2309–2319, 2001.

[24] K. Lengwehasatit, A. Ortega, "Scalable variable complexity approximate forward DCT," IEEE Transactions on Circuits and Systems for Video Technology, , vol.14, no.11, pp.1236-1248, 2004.

[25] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "Low-complexity 8 x 8 transform for image compression," Electron. Lett., vol. 44, no. 21, pp. 1249–1250, 2008.

[26] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A fast 8×8 transform for image compression," 2009 International Conference on Microelectronics (ICM), pp.74-77, 19-22 Dec. 2009.

[27] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A low-complexity parametric transform for image compression," in Proc. ISCAS, pp. 2145–2148. May 2011.

[28] R. J. Cintra and F. M. Bayer, "A DCT approximation for image compression," IEEE Signal Processing Letters, vol. 18, no. 10, pp. 579–582, Oct. 2011.

[29] F. M. Bayer and R. J. Cintra, "DCT-like transform for image compression requires 14 additions only," Electron. Lett., vol. 48, no. 15, pp. 919–921, 2012.

[30] Kaushal, V.; Garg, B.; Jaiswal, A.; Sharma, G.K., "Energy Aware Computation Driven Approximate DCT Architecture for Image Processing," in VLSI Design (VLSID), 2015 28th International Conference on , vol., no., pp.357-362, 3-7 Jan. 2015

[31] S.Cotofana, C. Lageweg, and S. Vassiliadis, "Addition Related Arithmetic Operations via Controlled Transport of Charge," IEEE Trans. Computers, vol. 54, no. 3, pp. 243-256, Mar. 2005

[32] Predictive Technology Model (PTM), http://ptm.asu.edu/.

[33] J,-F, Lin, Y,-T. Hwang, M.-H. Sheu, and C.-C. Ho," A Novel High-Speed and Energy Efficient 10-Transistor Full Adder Design" IEEE Transactions on Circuits and Systems—I: Regular Papers, vol. 54, no. 5, May 2007.

[34] Z. Pan and H. Bolouri, "High Speed Face Recognition Based on Discrete Cosine Transforms and Neural Networks," Tech. Rep., 1999, Science and Technology Research Centre, University of Hertfordshire.

[35] Z. Pan, R. Adams, and H. Bolouri, "Image Recognition using discrete cosine transforms as dimensionality reduction," in Proc. IEEE—EURASIP Workshop Nonlinear Signal and Image Process., 2001, pp. 149–154.

[36] J. Robinson and V. Kecman, "Combining support vector machine learning with the discrete cosine transform in image compression," IEEE Trans. Neural Netw., vol. 14, pp. 950–958, 2003.

[37] G. Mandyam, N. Ahmed, and N. Magotra, "Lossless image compression using the discrete cosine transform," J. Vis. Commun. Image Represent., vol. 8, no. 1, pp. 21–26, 1997.