Approximate DCT and Quantization Techniques for Energy-Constrained Image Sensors

Ming-Che Li*, *Student Member, IEEE*, Archisman Ghosh*, *Student Member, IEEE*, and Shreyas Sen, *Senior Member, IEEE.*
*Authors contributed equally

Abstract*—Recent expansions in multimedia devices for many applications, such as surveillance, self-driving cars, and health-care, gather enormous amounts of real-time images for processing and inference. The images are first compressed using compression schemes, like JPEG, before processing to reduce storage costs and additional power requirements for transmitting the captured data in this era of emerging ultra-wide-band communication and human-body communication. The JPEG algorithm realizes image compression using simplistic matrix manipulations, making it preferable for hardware implementations. Furthermore, due to inherent error resilience and imperceptibility in images, JPEG can be approximated to reduce the required computation/processing power and area. This work demonstrates the first end-to-end approximation computing-based optimization of JPEG hardware using i) an approximate division realized using bit-shift operators to reduce the complexity of the computationally intensive quantization block, ii) loop perforation, and iii) precision scaling on top of a multiplier-less fast DCT architecture to achieve an extremely energy-efficient JPEG compression unit which will be a perfect fit for power/bandwidth-limited scenario. Furthermore, a gradient descent-based heuristic composed of two conventional approximation strategies, i.e., Precision Scaling and Loop Perforation, is implemented for tuning the degree of approximation to trade off energy consumption with the quality degradation of the decoded image. The entire RTL (Register-Transfer Level) design is coded in Verilog HDL, synthesized using the industry-standard tool, and mapped to TSMC 65nm CMOS technology. and simulated using Cadence Spectre Simulator under 25°C, TT (Typical/Typical) corner. The approximate division approach in the quantization block achieved around 28% reduction in the active design area. The heuristic-based approximation technique combined with accelerator optimization achieves a significant energy reduction of 36% for a minimal image quality degradation of 2% SAD (Sum of Absolute Difference). Simulation results also show that the proposed architecture consumes 15uW at the DCT and quantization stages to compress a colored 480p image at 6fps.

***Index Terms*—Approximate computing, approximate divider, precision scaling, loop perforation, energy-efficient, image sensor, in-sensor analytics**

I. Introduction

USAGE of multimedia devices has expanded exponentially in recent years in every application, such as surveillance, self-driving cars, and healthcare. These sensors generate enormous amounts of data in images or videos that require processing and storage. As these sensors operate in an energy-constrained environment, as shown in Fig. 1, the collected images are often compressed at the source using conventional image compression techniques to mitigate the energy expended in transmission, processing, and storage.

Image compression algorithms can be categorized into two classes based on the error they introduced, i.e., lossy, and lossless compressions. Lossless compressions, like PNG (Portable Network Graphics) and TIFF (Tagged Image File Format), ensure perfect reconstruction of the image at the cost of reduced compression (usually 2:1 [1]). On the other hand, lossy schemes such as JPEG (Joint Photographics Experts Group), PGF (Progressive Graphics File), and JPEG2000 [2] provide higher compression while seeking to reconstruct a visually similar image with imperceptible degradation in quality.

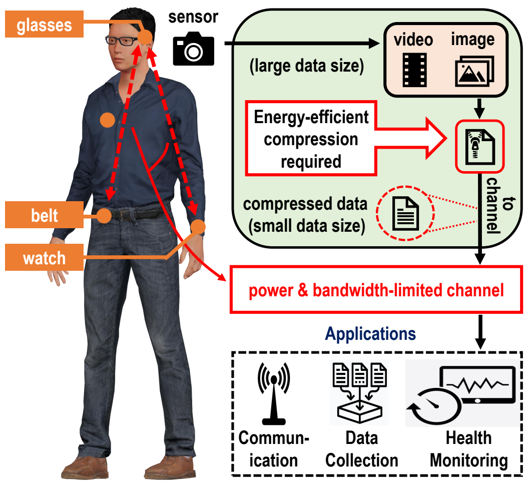

Fig. 1. Significance of energy-constrained compression hardware in a power & bandwidth-limited scenario.

Among these lossy compression schemes, JPEG is one of the most common image compression methods utilized in energy-constrained edge devices due to its lightweight nature and broad compatibility. Although JPEG2000 provides better compression efficiency and less image quality degradation at lower bit rates because of its compression algorithm [3] [4] [5] [6], JPEG has gained popularity over JPEG2000 due to its lower hardware requirements, which is a more critical factor for performing image compression in energy-constrained environments.

To date, approximate solutions for JPEG image compression are concentrated on algorithm or hardware levels. Previous works [7]–[10] present mathematical approximations on the Discrete Cosine Transform (DCT) matrix in which fractional coefficients are replaced with integers or negative powers of two, but no corresponding implementation on hardware is proposed in the literature. On the other hand, hardware-target approximation literature, such as [11] (performing bit truncation at DCT stage), [12] [13] [14] (using approximate

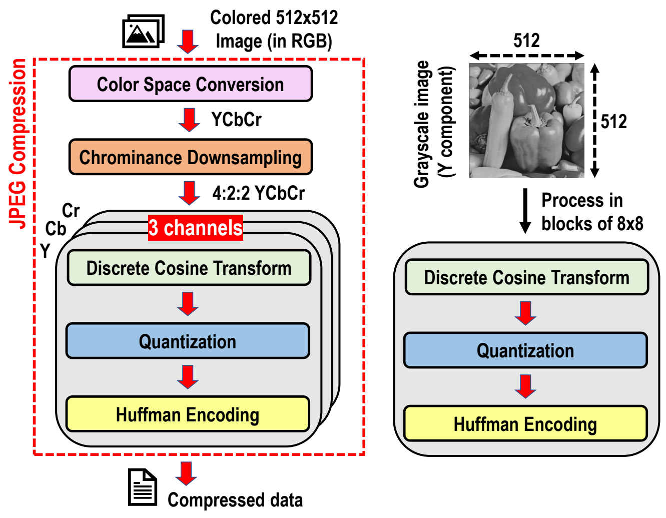

Fig. 2. Flowchart of standard JPEG compression.

adders), or [15] (replacing the quantization using approximate divider), only focus on the single stage optimization. They fail to produce an energy-optimal compression circuit that performs approximate computing on more than one of the JPEG compression stages.

In this work, we present an accelerated JPEG computing unit that incorporates an approximate quantization block and two well-known approximation techniques—loop perforation and precision scaling—on top of a multiplier-less fast DCT architecture to create a solution suitable for energy-constrained devices. A lightweight heuristic is also developed to estimate and fine-tune the degree of loop perforation and precision scaling without introducing significant degradation in the output image quality.

In brief, the proposed work has three-fold contributions:

1) We propose an approximate low-power area-efficient JPEG compression architecture focused on restructuring the computation-heavy quantization block. No such hardware-efficient image compression circuit has been reported that utilizes the bit shift operators-based division modules for optimizing the same to date.

2) A gradient descent-based heuristic is proposed that employs the commonly used loop perforation and precision scaling strategies to automatically configure the degree of approximation in the compression engine to reduce the energy required for processing while maintaining the quality of the decoded image within acceptable limits.

3) The reconfigurable architecture with optimized division modules and the implemented heuristics achieves a significant energy reduction of 36% for a minimal image quality degradation of SAD (2%).

The overall implementation achieves 28% area improvement with respect to the baseline design. It is important to note that our baseline design is already optimized using multiplier-less DCT architecture. Simulation results show that the proposed architecture consumes only 15uW power while compressing 480p images at a 6fps rate.

The rest of the paper is organized as follows. The JPEG compression technique is briefly discussed in Section II. Section III dives into the implemented approximation techniques along with the proposed heuristics that dynamically selects the optimum approximation knobs for operation. Simulation methodology is discussed in Section IV. Finally, we discuss the results in Section V, before concluding the paper in Section VII.

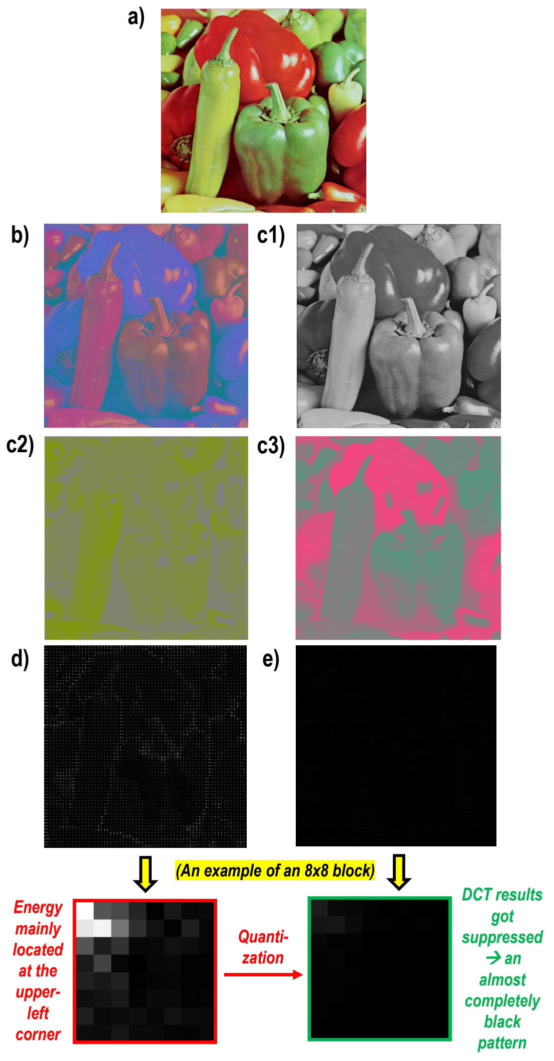
a)
b)
c1)
c2)
c3)
d)
e)
(An example of an 8x8 block)
Energy mainly located at the upper-left corner
Quanti-zation
DCT results got suppressed → an almost completely black pattern

Fig. 3. A visual example of the JPEG compression process: (a) colored (RGB format) image to be compressed. (b) colored image in YCbCr format. (c1) Y component represents the brightness of the original image. (c2)-(c3) Cb and Cr components represent the strength of blue and red signals of the original image, respectively. (d) the DCT (given by Eq. (1)) result of the Y component. (e) quantized result of (d).

II. Image Compression Technique

The Joint Photographics Experts Group developed the ISO (International Organization for Standardization) for the standard JPEG image compression [16] algorithm, which can be broken down into five essential stages. To compress one 512x512 pixels colored image, the image has to be run through the following stages, as shown in Fig. 2:

- Stage 1: Convert pixel values from RGB to YCbCr format. (Different from RGB color space, YCbCr is another color space where a colored image is described in terms of brightness (Y) and the chroma strength of blue (Cb) and red (Cr) signals.)

- Stage 2: Downsample the chrominance component (Cb and Cr) for every 2x2 pixel block.
- Stage 3: Discrete Cosine Transform (DCT) on 8x8 pixel block.
- Stage 4: Quantize DCT output matrix via element-wise division by quantization matrix.
- Stage 5: Perform Huffman Encoding on the zigzag traversal of the quantized matrix to keep low-frequency components at the beginning of the serial chain and insignificant high-frequency components at the end.

JPEG compression exploits the fact that the human eye is relatively insensitive to (1) chromaticity compared to luminosity and (2) high-frequency spatial changes in an image to compress an image. Because of property (1), the size of the data used to represent the color intensity of an image can be appropriately reduced without affecting the image's visual quality. This data size reduction is realized by the first two stages of JPEG compression, where an image is converted into the space of luminance and chrominance first, and its chrominance components are then sampled every 2 pixels in the horizontal direction while every luminance pixel is retained. On the other hand, high-frequency spatial changes in an image are suppressed in DCT, quantization, and Huffman encoding three stages. Note that starting from Stage 3, the compression is performed in the unit of 8x8 pixel blocks; moreover, three channels are required to process the compression of the original image's Y, Cb, and Cr components. Fig. 3 demonstrates the visual example of how an image is compressed.

After the first two stages, the input image is processed into chunks of pixel blocks. One block is represented as an 8x8 matrix $\mathbf{M}$. The **Discrete Cosine Transform** on $\mathbf{M}$ is given by the matrix multiplication:

$$\mathbf{D} = \mathbf{TMT}^{'} \tag{1}$$

where $\mathbf{D}$ is the DCT of $\mathbf{M}$, $\mathbf{T}$ is the transformation matrix, and $\mathbf{T}^{'}$ is transpose of $\mathbf{T}$. The transformation matrix $\mathbf{T}$ is defined as:

$$t_{ij} = \begin{cases} \frac{1}{\sqrt{8}}, & \text{if } i = 0 \\ \sqrt{\frac{2}{8}} \cdot \cos\frac{(2j+1)i\pi}{16}, & \text{if } i > 0 \end{cases} \tag{2}$$

where t_{ij} is the element in $\mathbf{T}$ and subscripts i, j are from 0 to 7 and represent the row position and the column position, respectively.

In this work, to build an energy-efficient JPEG compression circuit, we implement Eq. (1) not by the direct matrix multiplication method but by the **fast DCT (FDCT)** method [17]. Since Eq. (1) can be written as:

$$\mathbf{TMT}' = [(\mathbf{T})(\mathbf{TM})']', \tag{3}$$

the computation of $\mathbf{D}$ can be treated as 2 rounds of one-dimentional DCT (1D-DCT), as shown in Fig. 4. The 1D-DCT is the multiplication of the transformation matrix $\mathbf{T}$ and any given 8x1 vector $\mathbf{x}$. The computation of 1D-DCT can be mathematically 'accelerated' using the fast-architecture. The fast architecture with different types of butterfly units is shown in Fig. 5, which reduces the multiplications of $\mathbf{Tx}$ from 64 to 16. On the hardware level, an energy-efficient 1D-FDCT unit is achieved by adopting the multiplier-less approach [18], where four types of multiplications in 1D-FDCT (one scaler multiplication and three 2x2 matrix multiplications) are implemented by adders and shifters. Eq. (4a)(5a)(6a)(7a) delineate the precise mathematical expression for the four multiplications, with x and y representing the inputs and X and Y denoting the multiplication outputs. Eq. (4b)(5b)(6b)(7b) explicitly illustrate how these multiplications are executed through the shift add method. Fig. 6 portrays the associated hardware-level implementation, comprising solely adders, subtractors, and barrel-shifters. Our DCT core does not employ any general multiplier.

$Scaler\ multiplication$:

$$X = 0.707x \backsimeq \frac{181}{256}x = \frac{5(1-16)+256}{256}x \tag{4a}$$

$$\begin{aligned} X &= b >> 8 \\ b &= (a - a << 4) + (x << 8) \\ a &= x + (x << 2) \end{aligned} \tag{4b}$$

$Butterfly\ matrix\ multiplication\ (i)$:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0.9238 & 0.3836 \\ 0.3836 & -0.9238 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \backsimeq \frac{1}{2^9} \begin{bmatrix} 473 & 196 \\ 196 & -473 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{5a}$$

$$\begin{aligned} X &= \frac{1}{512}(473x + 196y) = d_x >> 9 \\ Y &= \frac{1}{512}(196x - 473y) = d_y >> 9 \\ d_x &= c_{xy} - (b_x << 6) + (y << 9) \\ d_y &= (c_{xy} << 3) - b_y \\ c_{xy} &= b_x + (a_{xy} << 5) \\ b_x &= x - (x << 3) + (y << 2) \\ b_y &= (a_{xy} << 2) + y \\ a_{xy} &= x - (y << 1) \end{aligned} \tag{5b}$$

$Butterfly\ matrix\ multiplication\ (ii)$:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0.9807 & 0.1951 \\ 0.1951 & -0.9807 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \backsimeq \frac{1}{2^8} \begin{bmatrix} 213 & 142 \\ 142 & -213 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6a}$$

$$\begin{aligned} X &= \frac{1}{256}(213x + 142y) = b_x >> 8 \\ Y &= \frac{1}{256}(142x - 213y) = b_y >> 8 \\ b_x &= -71a_x \\ b_y &= -71a_y \\ a_x &= x - (x << 2) - (y << 1) \\ a_y &= -(x << 1) + y + (y << 1) \end{aligned} \tag{6b}$$

$Butterfly\ matrix\ multiplication\ (iii)$:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0.8315 & 0.5556 \\ 0.5556 & -0.8315 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \backsimeq \frac{1}{2^8} \begin{bmatrix} 251 & 50 \\ 50 & -251 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7a}$$

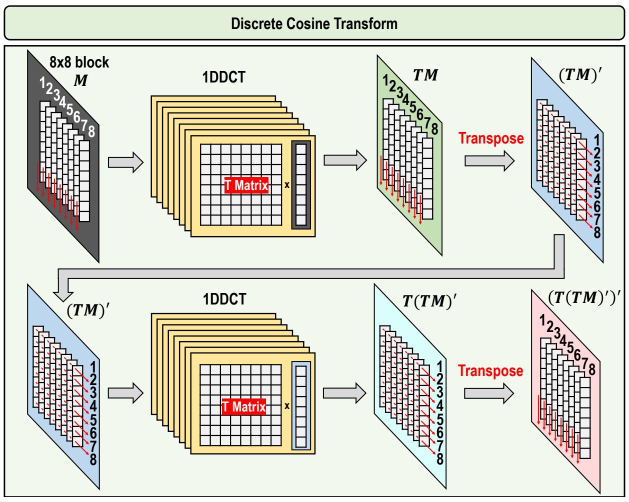

Fig. 4. Dividing 2-D DCT into 1-D DCT.

$$
\begin{aligned}
X &= \frac{1}{256}(251x+50y) = c_x >> 8 \\
Y &= \frac{1}{256}(50x-251y) = c_y >> 8 \\
c_x &= (x << 8) + b_x \\
c_y &= -(y << 8) + b_y \\
b_x &= a_x + (a_x << 2) \\
b_y &= a_y + (a_y << 2) \\
a_x &= -x + ((y + (y << 2)) >> 2) \\
a_y &= ((x + (x << 2)) >> 2) + y
\end{aligned}
\tag{7b}
$$

After the 8x8 pixel block is converted into the frequency domain, the DCT matrix $\mathbf{D}$ is next quantized by a designated quantization matrix $\mathbf{Q}$. The quantization stage generally performs element-wise division by directly instantiating divider blocks. The user can choose different quantization matrices based on the trade-off between the image quality and the compression level. The higher the quality level a $\mathbf{Q}$ matrix has, the less image compression will be. In other words, the information of the original image will not be discarded much, so when the image is reconstructed, higher image quality can be obtained. Typically, the quantization matrix with a quality level of 50, $\mathbf{Q_{50}}$, is used at this stage to achieve a good decompressed image quality and a decent compression ratio.

$$
\mathbf{Q_{50}} = \begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{bmatrix} \tag{8}
$$

If a higher image quality is needed, a quantization matrix with a quality level greater than 50 is necessitated. The required matrix is obtained by multiplying $\mathbf{Q_{50}}$ with a scaling factor of $(100 - \mathit{quality\ level})/50$ and then rounded and clipped so

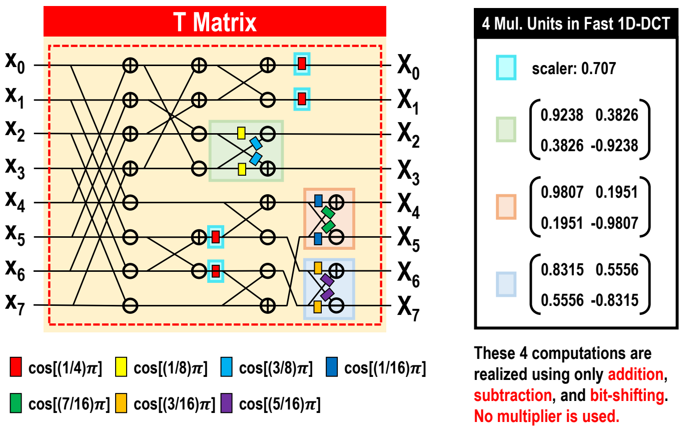

Fig. 5. 1D-FDCT: Butterfly diagram for 8-point 1D-DCT. Each colored small block represents a multiplication with a specific cosine factor.

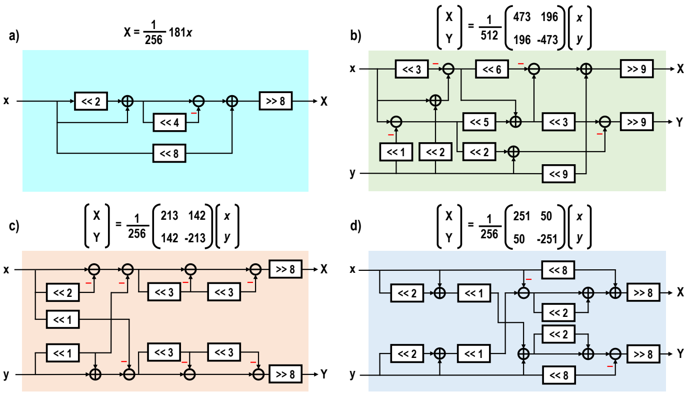
a)
X = 1/256 181x
<<2
<<4
<<8
>>8
b)
X Y = 1/512 (473 196 / 196 -473) x y
<<3
<<6
>>9
<<5
<<1
<<2
<<9
c)
X Y = 1/256 (213 142 / 142 -213) x y
d)
X Y = 1/256 (251 50 / 50 -251) x y

Fig. 6. Detailed multiplier-less implementation of (a) the scaling operation and (b)-(d) three 2x2 matrix multiplications in Fig. 5.

that all entry values are integers ranging from 1 to 255. For example, $\mathbf{Q_{90}}$, the quantization matrix with a quality level of 90, is given by

$$
\mathbf{Q_{90}} = \begin{bmatrix}
3 & 2 & 2 & 3 & 5 & 8 & 10 & 12 \\
2 & 2 & 3 & 4 & 5 & 12 & 12 & 11 \\
3 & 3 & 3 & 5 & 8 & 11 & 14 & 11 \\
3 & 3 & 4 & 6 & 10 & 17 & 16 & 12 \\
4 & 4 & 7 & 11 & 14 & 22 & 21 & 15 \\
5 & 7 & 11 & 13 & 16 & 12 & 23 & 18 \\
10 & 13 & 16 & 17 & 21 & 24 & 24 & 21 \\
14 & 18 & 19 & 20 & 22 & 20 & 20 & 20
\end{bmatrix}. \tag{9}
$$

The output of this stage is given by an 8x8 matrix $\mathbf{C}$, with $\mathbf{C} = \mathbf{D}\ ./\ \mathbf{Q}$, where $./$ represents the element-wise division operator. (We note that the entry values of $\mathbf{Q_{90}}$ are smaller than those of $\mathbf{Q_{50}}$, which implies $\mathbf{Q_{90}}$ will result in a $\mathbf{C}$ with larger entry values than $\mathbf{Q_{50}}$ will. That is, there will be more information for the image's reconstruction and higher reconstructed image quality can be obtained.)

Finally, the quantized result is compressed into a bit stream using Huffman Encoding. The elements $\mathbf{C}$ are concatenated into a 1D vector based on the zig-zag traversal. After concatenating the bits in this traversal as an input bit stream, a binary tree of the most common N-bit groups is created where a traversal of the encoding tree in the left or right directions maps to a binary bit (0 or 1). This allows an N-bit chunk of zeros (most common) to map to only a single bit after Huffman Encoding. After encoding, a compressed bit stream is created and sent to the communication channel.

III. Approximation Techniques

To date, the approximate JPEG compression technique has been explored by using only bit truncation [11], dynamic bit width reduction in DCT operation [13], or using an approximate adder [12]. However, we observed that the quantization block realized using standard division algorithms consumes high power while occupying a considerable silicon area.

For the first time, we explored an approximate quantization block by updating the Q-matrix to enable divisions with bit-shift operations, eliminating the need for high-budget standard division blocks, thereby saving energy and reducing silicon area. Conventional approximation strategies, like loop perforation and precision scaling, are also explored in this work. In addition to the approximate quantization block, we have proposed a heuristic-based approach to select the optimal configuration between loop perforation and precision scaling for a given quality requirement.

A. Approximate Quantization

A common approach to reducing the power of the Q block is to replace the standard division $\frac{A}{B}$ with multiplication using $A \cdot \frac{1}{B}$ using techniques like Taylor Series expansion to approximate $\frac{1}{B}$ [19], [20] or reducing the width of operation in division block [21]. These methods, however, require one or more multipliers, which demand relatively higher energy.

By observation, quantization matrix $\mathbf{Q}$ can be replaced with approximated quantization matrix $\mathbf{Q'}$ by converting each element of the $\mathbf{Q}$ matrix to the power of 2 so that the division operation can be implemented via bit shifting. For example, $\mathbf{Q_{50}}$ can be approximated as:

$$\mathbf{Q'_{50}} = \begin{bmatrix} 16 & 8 & 8 & 16 & 16 & 32 & 32 & 32 \\ 8 & 8 & 8 & 16 & 16 & 32 & 32 & 32 \\ 8 & 8 & 16 & 16 & 32 & 32 & 64 & 32 \\ 8 & 16 & 16 & 16 & 32 & 64 & 64 & 32 \\ 16 & 16 & 32 & 32 & 64 & 64 & 64 & 64 \\ 16 & 32 & 32 & 32 & 64 & 64 & 64 & 64 \\ 32 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \end{bmatrix}. \tag{10}$$

Similarly, $\mathbf{Q_{90}}$ can be approximated as:

$$\mathbf{Q'_{90}} = \begin{bmatrix} 2 & 2 & 2 & 2 & 4 & 8 & 8 & 8 \\ 2 & 2 & 2 & 4 & 4 & 8 & 8 & 8 \\ 2 & 2 & 2 & 4 & 8 & 8 & 8 & 8 \\ 2 & 2 & 4 & 4 & 8 & 16 & 16 & 8 \\ 4 & 4 & 4 & 8 & 8 & 16 & 16 & 8 \\ 4 & 4 & 8 & 8 & 16 & 8 & 16 & 16 \\ 8 & 8 & 16 & 16 & 16 & 16 & 16 & 16 \\ 8 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix}. \tag{11}$$

This approach introduces some errors due to approximation but reduces the divider power consumption because of the simplicity of bit shifting. Note that instead of being converted up, the elements in the original matrix are converted down to the nearest power of 2 to retain higher image quality. The general mathematical behavior of the proposed approximate technique is described as follows: Given a quantization matrix $\mathbf{Q}$, for each element q_{ij} in $\mathbf{Q}$, we first locate q_{ij} using an integer s_{ij} which satisfies:

$$2^{s_{ij}} \leq q_{ij} < 2^{s_{ij}+1} \tag{12}$$

The corresponding approximated element in $\mathbf{Q'}$, q'_{ij}, is then constructed by:

$$q'_{ij} = 2^{s_{ij}} \tag{13}$$

If all elements in $\mathbf{Q'}$ are considered, Eq. (7) can be further extended to a matrix form:

$$\mathbf{Q'} = 2.^\wedge\mathbf{S} \tag{14}$$

where $.^\wedge$ is a element-wise power operator and $\mathbf{S}$ is an 8x8 matrix with its element s_{ij} representing the exponent part of q'_{ij}. Since $1 \leq q_{ij} \leq 255$, we can obtain $0 \leq s_{ij} \leq 7$ according to Eq. (6); therefore, s_{ij} can be represented using only 3 bits. Using this approximate technique, the quantization circuit in the JPEG compression circuit only needs to take 192 bits (3 bits x 64 elements) as the input, while the conventional division-based quantization circuit requires 512 bits (8 bits x 64 elements). At the hardware level, Eq. (12)(13) can be implemented by an 8-to-3 priority encoder (Fig. 7(a)). For an 8-bit input $q_{ij}[7{:}0]$, an 8-to-3 priority encoder can locate the first bit appearing from the MSB side and output that particular location by a 3-bit signal $s_{ij}[2{:}0]$. The quantization is then performed by a bit shifter, with $s_{ij}[2{:}0]$ being the shifting amount. Therefore, our proposed architecture realizes the approximated element-wise division operation through an 8-to-3 priority encoder and a barrel-shifter, as shown in Fig. 7(b). Note that at the time of decoding, the same quantization matrix $\mathbf{Q'}$ must be used for better reconstruction of the image.

B. Precision Scaling

Precision scaling, or bit truncation, alleviates the computational load of image compression by reducing the data width of the input image. LSB (Least Significant Bit) truncation is realized in this paper. The size of data reduction, *truncation level* B_j, has to be specified before image compression begins. The truncated pixel block, $\mathbf{M_{tr}}$, can be described in terms of the original pixel block $\mathbf{M}$ and truncation level B_j as

$$\mathbf{M_{tr}} = round\{\frac{1}{2^{B_j}}\mathbf{M}\} \tag{15}$$

The precision scaling technique allows the DCT and quantization to function with fewer bits overhead. Note that bit truncation is implemented uniformly throughout the data path of the hardware accelerator. Each data-path component (adders, multipliers, etc.) can be equipped to modify the operational bit-width by introducing a bit-wise clock gating in each of them, thereby reducing energy consumption.

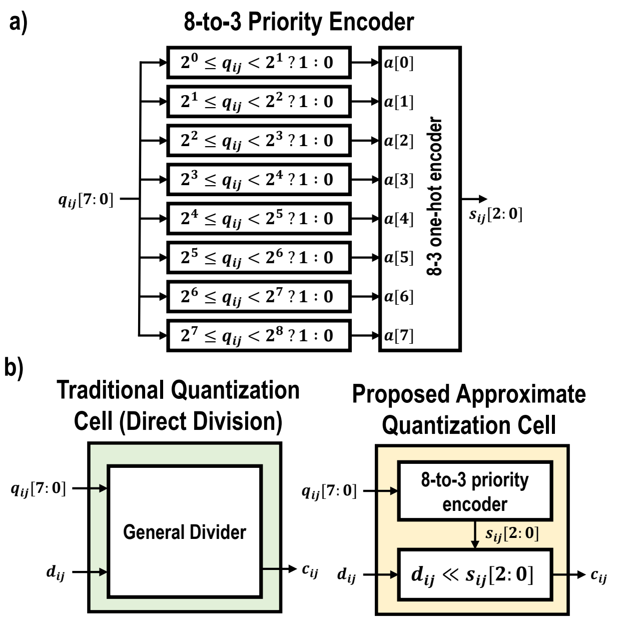
a)
8-to-3 Priority Encoder
$2^0 \le q_{ij} < 2^1\, ?\, 1 : 0$
$2^1 \le q_{ij} < 2^2\, ?\, 1 : 0$
$2^2 \le q_{ij} < 2^3\, ?\, 1 : 0$
$2^3 \le q_{ij} < 2^4\, ?\, 1 : 0$
$2^4 \le q_{ij} < 2^5\, ?\, 1 : 0$
$2^5 \le q_{ij} < 2^6\, ?\, 1 : 0$
$2^6 \le q_{ij} < 2^7\, ?\, 1 : 0$
$2^7 \le q_{ij} < 2^8\, ?\, 1 : 0$
$q_{ij}[7:0]$
$a[0]$ $a[1]$ $a[2]$ $a[3]$ $a[4]$ $a[5]$ $a[6]$ $a[7]$
8-3 one-hot encoder
$s_{ij}[2:0]$
b)
Traditional Quantization Cell (Direct Division)
$q_{ij}[7:0]$
d_{ij}
General Divider
c_{ij}
Proposed Approximate Quantization Cell
$q_{ij}[7:0]$
8-to-3 priority encoder
$s_{ij}[2:0]$
d_{ij}
$d_{ij} \ll s_{ij}[2:0]$
c_{ij}

Fig. 7. Hardware implementation of the proposed approximate quantization method: (a) An 8-to-3 priority encoder. (b) The proposed element-wise division cell comprises an 8-to-3 priority encoder and a barrel-shifter.

C. Loop Perforation

Loop perforation, or loop skipping, takes advantage of spatial redundancies in an image. This technique involves bypassing the compression process for the current pixel block if it closely resembles the preceding one. This results in a substantial decrease in the energy expended for computation while preserving an acceptable level of degradation in the encoded image quality.

This work implements the loop skipping function by first asserting the loop skip threshold $L_i = i$ (i is a non-negative integer), representing the error tolerance ε with $\varepsilon = 5i$. To determine whether a pixel block should be sent into the JPEG compression computation core, the pixel block is compared with the previously processed pixel block by checking if all pixels in two blocks are within the error tolerance. The exact step-by-step operation is illustrated in Algorithm 1. If a pixel block satisfies the loop skipping criteria, the JPEG compression circuit will disable the computation core and directly output the computation result of the previously processed pixel block.

Fig. 8(a) shows the additional hardware cost incurred to perform loop skipping, where registers for storing the previous block and its corresponding results, a similarity checker, and related selection logic (MUX) are added to the original JPEG core. Fig. 8(b)(c) provide insight into the data flow and circuit operation under conditions where the similarity of two blocks is detected or not. Fig. 8(b) depicts the situation where no similarity is detected. If two neighboring blocks are not similar enough, the similarity check logic will generate a FALSE signal, writing 64 pixels of the current block into the previous block register and saving the current block compressed results for the next cycle comparison. On the other hand, if the similarity is detected, a TRUE signal will be sent out from the similarity check logic, which disables the JPEG core and outputs the computation results directly from the compressed result registers. This scenario is illustrated in Fig. 8(c).

The overall power savings remain substantial despite the need for additional hardware resources at the circuit level to implement loop perforation, such as the logic determining the similarity of the pixel blocks and registers storing the previously processed pixel block and its corresponding result. This is primarily due to the disabling of the DCT unit, a significantly more power-consuming block. This point will be further discussed in Section V-E.

By increasing the loop skipping levels, higher energy savings are achieved at the expense of degraded image quality. In this work, we have implemented loop skipping in the software and the hardware that performs the desired operations and generates the required control signals before the acceleration.

Algorithm 1: Loop skipping logic

```
Input: the on-trial 8x8 block $\mathbf{M_{in}}$, the latest-processed
       block $\mathbf{M_l}$ and its JPEG compression result $\mathbf{C_l}$,
       and the error tolerance $\varepsilon$
Output: the new latest-processed block $\mathbf{M_{out}}$ and its
        corresponding result $\mathbf{C_{out}}$
for ($i = 0; i < 8; i = i + 1$) do
    for ($j = 0; j < 8; j = j + 1$) do
        ceiling = $\min\{m_{l,ij} + \varepsilon, 127\}$;
        floor = $\max\{m_{l,ij} - \varepsilon, -128\}$;
        if (($m_{in,ij}$ > ceiling) or ($m_{in,ij}$ < floor)) then
            $\mathbf{M_{out}} = \mathbf{M_{in}}$;
            $\mathbf{C_{out}} = JpegCompression(\mathbf{M_{in}})$;
            return $\mathbf{M_{out}}$, $\mathbf{C_{out}}$;
$\mathbf{M_{out}} = \mathbf{M_l}$; $\mathbf{C_{out}} = \mathbf{C_l}$;
return $\mathbf{M_{out}}$, $\mathbf{C_{out}}$;
```

D. Dynamic selection of optimum approximation technique

To maximize the energy savings from approximate computing, we propose to combine the *loop skipping* and *precision scaling* techniques. The combination of both strategies performs better than either standalone scheme.

Algorithm 2: To extract the *Q-E* characteristics for individual approximation technique

```
Input: Set of required image qualities: $Q[0 : N-1]$
Output: Set of quality knob configuration: $k[0 : N-1]$
        and energy consumption: $E[0 : N-1]$
        corresponding to $Q[0 : N-1]$
for ($i = 0; i < N; i = i + 1$) do
    m = 0;
    while (($k[m]$) $\ge Q[i]$) do
        m = m+1;
    $E[i] = E[(k[m-1])]$; $k[i] = k[m-1]$;
return $k, Q, E$
```

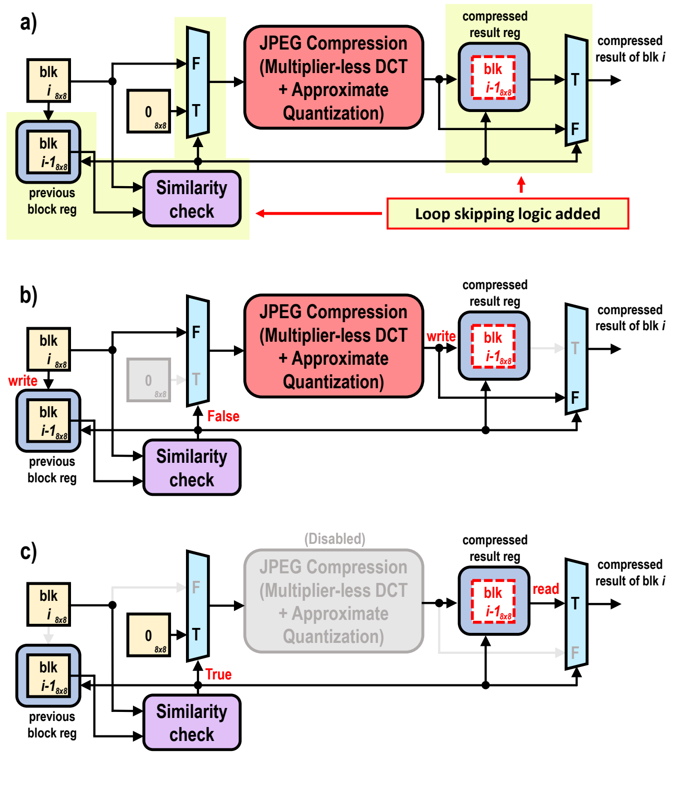
a)
b)
c)
blk i
blk i-1
previous block reg
Similarity check
JPEG Compression (Multiplier-less DCT + Approximate Quantization)
compressed result reg
compressed result of blk i
Loop skipping logic added
write
read
False
True
(Disabled)

Fig. 8. Hardware realization of loop perforation: a) additional registers and logic added, b) data flow when the similarity of neighboring blocks is not detected, and c) data flow when the similarity of neighboring blocks is detected.

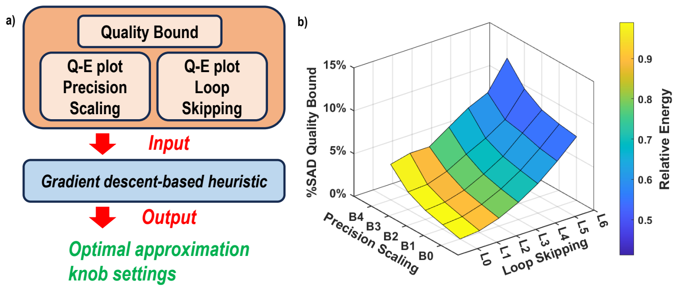

Fig. 9. Gradient Descent Algorithm: (a) block diagram and (b) 3D Quality vs. Energy plot.

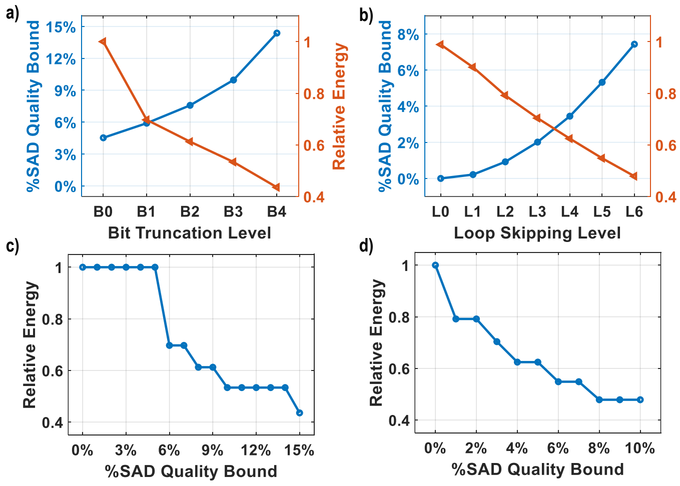

Fig. 10. (a)-(b): Normalized energy consumption vs SAD degradation bound for (a) Bit Truncation and (b) Loop skipping. (c)-(d): Individual E plots for (c) Bit Truncation and (d) Loop Skipping generated using Algorithm 2.

A gradient descent-based heuristic algorithm determines the optimal approximation degrees of bit truncation and loop skipping using the Quality-Energy (Q-E) plots of individual approximation strategies, as suggested by Fig. 9(a). To quantify the effect of approximation on the quality of the image, SAD (Sum of Absolute Differences) is chosen as a performance metric, which is defined as the ratio of the sum of absolute differences in pixel values between the generated and the reference image to the sum of pixel values in the reference image. Note that %SAD degradation has shown a good correlation to other metrics such as PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity) and is used for its easy computation in the heuristics and lower overhead. Also, note that the quality of the image is inversely proportional to this metric. However, we use SSIM and PSNR while evaluating the approximation techniques, as those metrics are well-accepted in the image processing community.

Algorithm 2 obtains the individual Q-E plots that provide insights into the degradation of the quality of the decoded images for benefits in relative energy for different approximation scenarios. For a particular output image quality bound, the quality knobs $B_0 - B_4$ and $L_0 - L_6$, along with the relative energy savings, are found by the online image gallery of the Computer Vision Group of the University of Granada [22]. Fig. 10 shows the extracted plots for loop perforation and precision scaling.

Algorithm 3 employs a gradient descent-based optimization search using these extracted plots to provide an overall Quality vs. Energy for the combined strategy. We vary the loop perforation categories (L_i) and bit truncation levels (B_j) to obtain the optimum settings for a particular output quality bound (Q_A). In other words, we are searching for the optimal solution $(\hat{B}_i, \hat{L}_i)$ such that

$$\begin{aligned} \text{minimize} \quad & E(B_i, L_i) \\ \text{subject to} \quad & Q(B_i, L_i) \leq Q_A \\ & Q(B_i, L_i) \geq 0 \\ & B_i \geq 0,\ L_i \geq 0 \end{aligned} \tag{16}$$

where $E(B_i, L_i)$ and $Q(B_i, L_i)$ represent the relative energy and %SAD degradation under particular B_i and L_i, respectively. The convexity of the problem can be justified as follows: First, $Q(B_i, L_i)$ is monotonically increasing in each dimension because higher bit truncation or loop skipping levels result in higher %SAD degradation. Second, since the relative energy and quality degradation are two inversely related variables, to minimize $E(B_i, L_i)$ is equivalent to maximizing $Q(B_i, L_i)$. With these two properties, Eq. (16) can be treated as a maximization problem constrained in the first octant of the 3-D space expanded by B_i, L_i, and $Q(B_i, L_i)$, where the objective function is monotonically increasing in the direction of B_i and L_i. As a result, the convexity is assured, and it is feasible to apply a gradient descent algorithm to find the optimal.

The controller implementing this heuristic, realized in software code, automatically configures the degree of loop perforation and bit truncation, L_i and B_j, respectively, by moving in the direction of the steepest gradient of the ratio of energy savings to quality degradation resulting from the variation in each degree of the approximation knobs. Fig. 9(b) shows the 3D Q-E plot for different L_i and B_j, along with the relative energy required for processing. For a specified quality degradation bound, our proposed gradient descent algorithm uses the 3D plot and selects the best (B_i, L_j) combination that results in the lowest energy configuration according to the color bar on the right according to Algorithm 3.

Algorithm 3: Gradient descent determines the optimal approximation degrees for a given quality bound

```
Input: Output quality bound: $Q_A$,
Quality vs. Energy ($Q$-$E$) curves for loop perforation
and precision scaling ($Q_l$-$E_l$) and ($Q_t$-$E_t$), respectively
Output: Optimal approximation knob settings ($i$, $j$)
        according to $Q_A$
1  Initialize: $i = j = 0$, $Q = 1$, $E = 1$
2  while ($Q \geq Q_A$) do
3      $E_{l\Delta}$=$E_l[i]$-$E_l[i+1]$; $Q_{l\Delta}$=$Q$-$Q_l[i+1]$;
4      $E_{t\Delta}$=$E_t[j]$-$E_t[j+1]$; $Q_{t\Delta}$=$Q$-$Q_t[j+1]$;
5      if ($\frac{Q_{l\Delta}}{E_{l\Delta}} \geq \frac{Q_{t\Delta}}{E_{t\Delta}}$) then
6          if ($Q_t[j+1] \leq Q_A$) then
7              $E = E - E_{t\Delta}$; $Q = Q - Q_{t\Delta}$; $j = j + 1$;
8          else if ($Q_l[i+1] \leq Q_A$) then
9              $E = E - E_{l\Delta}$; $Q = Q - Q_{l\Delta}$; $i = i + 1$;
10     else
11         if ($Q_l[i+1] \leq Q_A$) then
12             $E = E - E_{l\Delta}$; $Q = Q - Q_{l\Delta}$; $i = i + 1$;
13         else if ($Q_t[j+1] \leq Q_A$) then
14             $E = E - E_{t\Delta}$; $Q = Q - Q_{t\Delta}$; $j = j + 1$;
15 return $i$, $j$;
```

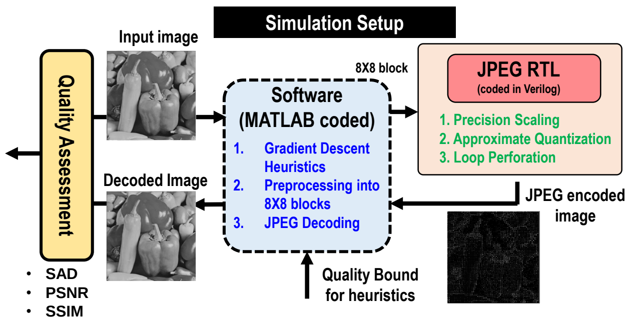

Fig. 11. Simulation setup.

IV. Simulation Methodology

Fig. 11 depicts the simulation setup used for testing the functionality of the hardware. Our simulation is conducted over an image dataset from the gallery of the Computer Vision Group of the University of Granada [22], where many representative images in the field of image processing, such as Baboon, Boat, Barbara, Pirate, Bridge, and Airplane, are included. The input image is reduced to chunks of 8×8 matrix in MATLAB before feeding to the design under test (DUT). The MATLAB code runs the gradient descent-based heuristic algorithm to estimate the optimal approximation knobs for a particular input quality bound (SAD). The software also decides the degree of precision scaling to be realized and configures the hardware by clock gating the required bits throughout the accelerator. A Verilog test bench is used to convey the appropriate degree of truncation and the pre-processed image (in a text file) for simulation. The accelerator performs the JPEG encoding on the input image and writes the output processed image in a separate text file, which is then reconstructed through inverse quantization and inverse DCT in the software. Inverse quantization is an operation of element-wise multiplication, which is given by:

$$\mathbf{R} = \mathbf{C} \odot \mathbf{Q} \tag{17}$$

where $\odot$ is the element-wise multiplication operator, and $\mathbf{R}$ is the result of inverse quantization.

In general, the $\mathbf{Q}$ used in this step should be the same one as used in the encoding process. However, to give a more comprehensive analysis of the effect of introducing an approximated quantization matrix, we also analyze the case where the approximated matrix $\mathbf{Q}'$ is used in encoding while the unmodified one $\mathbf{Q}$ is still used for inverse quantization.

Inverse DCT is given by the transformation of:

$$\mathbf{N} = \mathbf{T}^{'}\mathbf{R}\mathbf{T} \tag{18}$$

where $\mathbf{N}$ is the result of inverse DCT and $\mathbf{T}$ is given by Eq. (2). The reconstructed image can be obtained after rounding $\mathbf{N}$'s all entries. In the end, performance evaluation and quality assessment are conducted based on the reconstructed results.

To validate the accelerator's functionality, we use in-built MATLAB-based JPEG compression and compare it with hardware output. The JPEG RTL is synthesized using Synopsys Design Compiler, mapped to TSMC 65nm standard cell library. The functionality of the extracted netlist is re-validated. All the RTL simulations are performed using the Cadence NC-Verilog simulator. The design area and power values are provided from the post-synthesis simulation results. Note that results from Spice simulations (done in Cadence Virtuoso) are utilized as they provide precise energy numbers.

V. Results

In this section, we first discuss the effect of individual approximation techniques on the overall performance of JPEG compression hardware. The performance of the combined approximation strategy that dynamically tunes the configuration of the constituent techniques is also shown. Note that system-level hardware performances like power and area are not reported in previous related literature, such as [23] (which applies bit-truncation and loop peroration in JPEG compression) or [24] (which works on the approximation of DCT hardware).

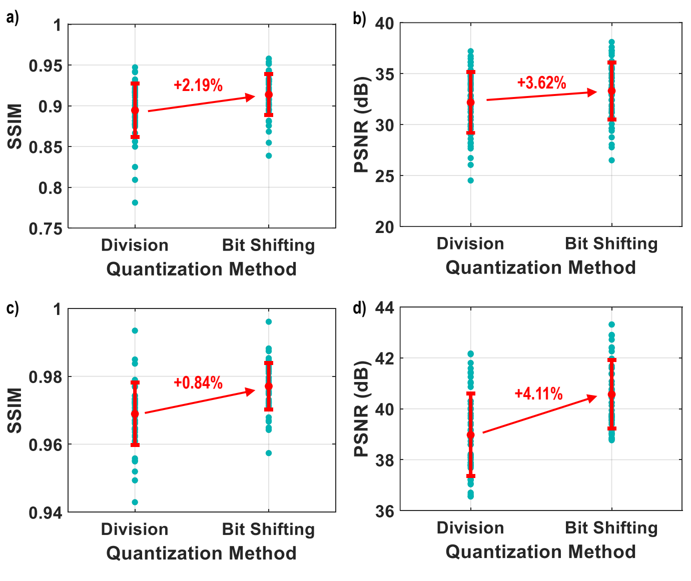

Fig. 12. Effect of quantization block approximation on image quality: (a)-(b) SSIM and PSNR comparison for Q50, and (c)-(d) SSIM and PSNR comparison for Q90.

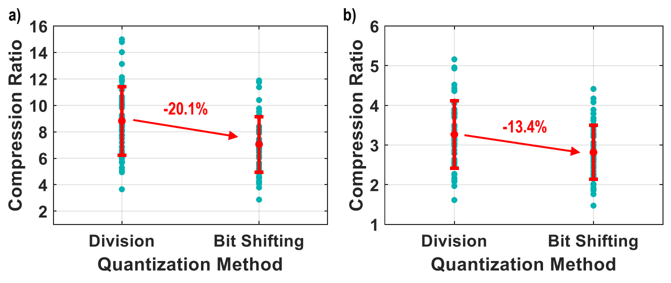

Fig. 13. Effect of quantization block approximation on compression ratio: (a) on Q50 and (b) on Q90.

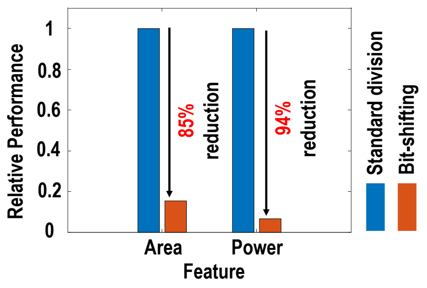

Fig. 14. Hardware-level area and power improvement based on approximate quantization: bit-shifting-based quantization vs. division-based quantization.

However, this work provides the power and area numbers for the most optimal design, including the DCT, approximate quantization block, and loop-skipping circuitry from the post-synthesis Spice simulations. The quality of the images is evaluated in terms of SSIM, PSNR, and SAD as discussed earlier in Section III.

A. Approximate Quantization

1) Case I: images reconstructed by the corresponding encoding quantization matrix (Eq. (10)(11)): The scatter plots from Fig. 12 and the corresponding statistic value in Table I together show the change in the SSIM and PSNR of the reconstructed images because of the use of an approximation matrix. It is observed that the reconstructed images encoded using bit-shifting-based quantization demonstrate better quality (higher SSIM and higher PSNR) than the reconstructed images encoded using standard division-based quantization for quality levels 50 and 90. This is because every element in the quantization matrix is down-approximated to its closest power of 2, resulting in a new quantization matrix whose quality factor is higher than the original's. This result, however, entails a reduction in the compression ratio of the compressed image.

Fig. 13 shows the effect of the approximated quantization method on compression ratio, where the 20.1% and 13.4% compression ratio decline are observed when approximated Q50 and Q90 are adopted, respectively. Detailed statistic values of Fig. 13 are provided in Table II. Although there are reductions in compression ratio, the advantages of using approximated quantization matrices are evident once the hardware-level implementation is considered. Fig. 14 enlightens the benefits in power and area using the proposed quantization scheme, achieving 85% reduction in area and 94% power savings for conventional division-based quantization block.

2) Case II: images reconstructed by the unmodified quantization matrix (Eq. (8)(9)): One should decode the compressed image with the same quantization matrix used in the encoding stage for better image quality when reconstructing an image. However, since the approximate quantization circuit is only applied in the encoding end in this work, we assume that the decoding end may still use the unmodified quantization matrix to reconstruct images. The discussion about using different quantization matrices for encoding and decoding is thus presented in Fig. 15, which provides the image quality comparison between images reconstructed with the modified (approximated) $\mathbf{Q}$ matrix and with the original (standard) one for $\mathbf{Q_{50}}$ and $\mathbf{Q_{90}}$. Using the standard matrix $\mathbf{Q_{50}}$ to reconstruct the image encoded by the approximated matrix $\mathbf{Q'_{50}}$ results in 4.94% SSIM degradation, as shown in Fig. 15(a). However, in the case of $\mathbf{Q_{90}}$, using the standard $\mathbf{Q_{90}}$ to reconstruct images induces more SSIM degradation than using the standard $\mathbf{Q_{50}}$, as shown in Fig. 15(b), where 22.35% SSIM degradation is presented.

This result stems from the fact that the first entry in Q50 is originally a 2's power ($\mathbf{Q_{50}}$(0,0) = 16), as is not the case for Q90 ($\mathbf{Q_{90}}$(0,0) = 3). For the scenario where $\mathbf{Q'_{90}}$ is used in encoding while $\mathbf{Q_{90}}$ is used in decoding, different divisors for the first entry of the DCT coefficient block (also known as the DC coefficient) are used in the quantization ($\mathbf{Q'_{90}}$(0,0) = 2), and inverse quantization ($\mathbf{Q_{90}}$(0,0) = 3). This discrepancy corrupts the reconstructed value of the DC coefficient at the step of inverse quantization. Since for still images, most of the energy is located in the low-frequency area [25], the corrupted DC coefficient will then result in severe image degradation after inverse DCT is performed. Note that this problem could

TABLE I
STATISTICS OF FIG. 12: IMAGE QUALITY UNDER DIFFERENT QUANTIZATION SCHEMES.

Quality Level		SSIM				PSNR			
		mean	std	max	min	mean	std	max	min
Q50	Division	0.8944	0.0327	0.9472	0.7811	32.17	2.99	37.20	24.51
	Bit shifting	0.9137	0.0252	0.9578	0.8387	33.30	2.80	38.10	26.49
	% Increase	2.19 %	1.22 %	7.37 %	0.99 %	3.62 %	1.24 %	8.09 %	2.07 %
Q90	Division	0.9689	0.0092	0.9934	0.9429	38.98	1.62	42.17	36.56
	Bit shifting	0.9770	0.0069	0.9961	0.9574	40.57	1.35	43.31	38.77
	% Increase	0.84 %	0.3 %	1.56 %	0.26 %	4.11 %	0.94 %	6.14 %	2.63 %

TABLE II
STATISTICS OF FIG. 13: COMPRESSION RATIO UNDER DIFFERENT QUANTIZATION SCHEMES.

Q Matrix		Compression Ratio			
		mean	std	max	min
Q50	Division	8.82	2.6	14.99	3.65
	Bit shifting	7.06	2.1	11.88	2.87
	Degradation	20.1 %	1.85 %	17.13 %	26.4 %
Q90	Division	3.27	0.89	5.16	1.61
	Bit shifting	2.82	0.68	4.41	1.47
	Degradation	13.4 %	1.94 %	8.77 %	17.34 %

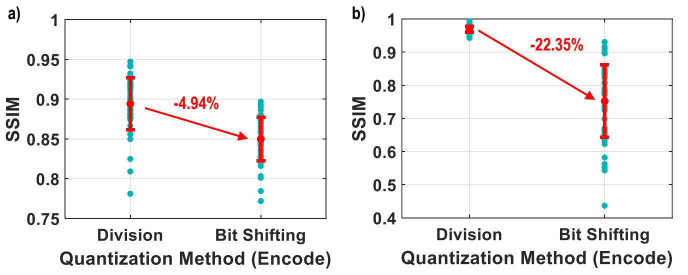

Fig. 15. Comparison of reconstructed image quality between the image decoded by the standard Q and by the modified Q for (a) Q50 and (b) Q90.

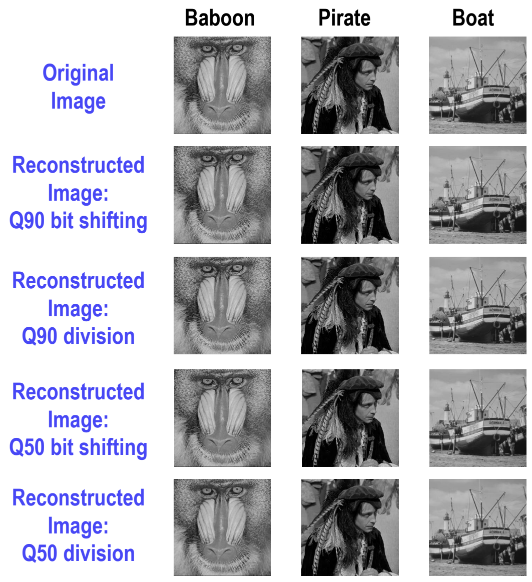
Baboon
Pirate
Boat
Original Image
Reconstructed Image: Q90 bit shifting
Reconstructed Image: Q90 division
Reconstructed Image: Q50 bit shifting
Reconstructed Image: Q50 division

Fig. 16. Reconstructed images using approximate division block vs. standard division block with two different quality levels 50 and 90.

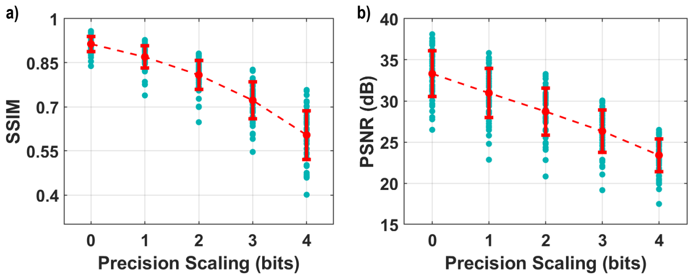

Fig. 17. Effect of precision scaling: (a) SSIM vs. precision scaling level, and (b) PSNR vs. precision scaling level.

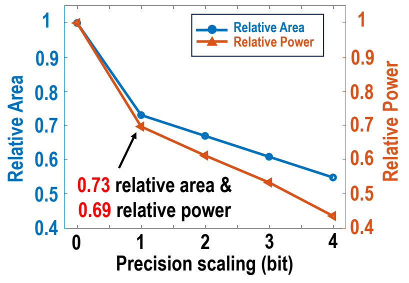

Fig. 18. Hardware-level area and power improvement based on different precision scaling levels.

be addressed by keeping the first entry in the standard $\mathbf{Q_{90}}$ matrix unmodified and designing a multiplier-less divider. For example, the element-wise division for $\mathbf{Q_{90}}$'s first entry (quantizing a number by 3) can be approximately implemented as 1/4+1/8-1/16+1/32 (0.334). Thus, this can be implemented just by addition and subtraction along with bit-shift operation.

Lastly, we present subjective analysis on three images (Baboon, Pirate, and Boat) selected from the image dataset. Fig. 16 shows the reconstructed images using different quantization methods and levels.

B. *Precision scaling*

Fig. 17 depicts the effect of precision scaling on the reconstructed image quality. Detailed statistics of the quality degradation is reported in Table III. With one-bit truncation, the quality of the reconstructed images degrades slightly; only 4.82% SSIM degradation and 7.05% PSNR degradation are observed according to the simulation results over the dataset. Hardware-wise benefit resulting from the bit truncation technique is illustrated in Fig. 18. The JPEG compression circuit with 1-bit truncation consumes around 30% less power and area than the circuit with no data bit width modification.

TABLE III
STATISTICS OF FIG. 17: IMAGE QUALITY UNDER DIFFERENT BIT TRUNCATION LEVELS.

Bits truncated	SSIM				PSNR			
	mean	std	max	min	mean	std	max	min
0	0.9137	0.0252	0.9578	0.8387	33.31	2.80	38.11	26.49
1	0.8697	0.0377	0.9275	0.7388	30.96	3.00	35.83	22.86
2	0.8084	0.0487	0.8814	0.6477	28.71	2.86	33.27	20.83
3	0.7221	0.0622	0.8264	0.5466	28.33	2.56	30.05	19.16
4	0.6041	0.0828	0.7574	0.4009	23.40	1.98	26.49	17.49

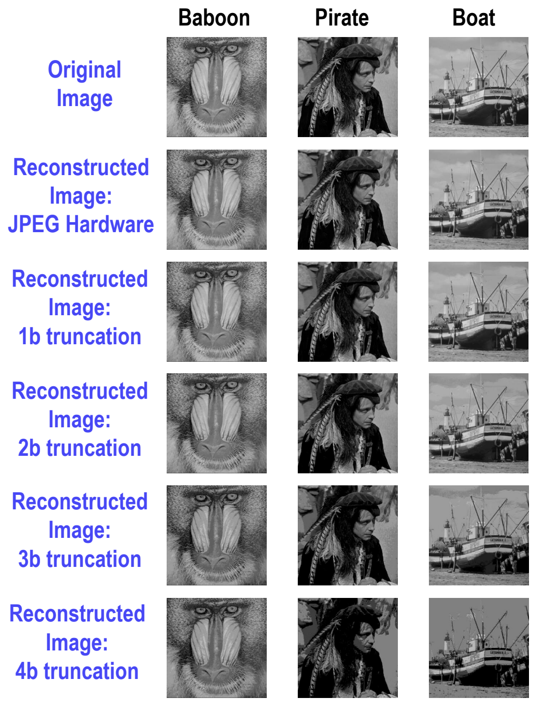
Baboon
Pirate
Boat
Original Image
Reconstructed Image: JPEG Hardware
Reconstructed Image: 1b truncation
Reconstructed Image: 2b truncation
Reconstructed Image: 3b truncation
Reconstructed Image: 4b truncation

Fig. 19. Reconstructed images for different precision scaling (bit truncation) levels.

Fig. 19 shows the subjective analysis on three selective images, Baboon, Pirate, and Boat, under different truncation levels (0 to 4). The quality of the reconstructed images degrades significantly under bit truncation levels 3 and 4. It can be seen from the images in the last two rows of Fig. 19 that they are heavily blurred compared to the original ones.

C. Loop Perforation

Fig. 20(a) and Table IV present the simulation results of how much energy is saved based on a particular loop-skipping level, and Fig. 20(b) shows the correlation coefficient between the energy saved and the image *homogeneity* under different loop-skipping levels. (*Homogeneity* is one of the Haralick textural features developed in [26], which is an indicator to describe pixel discrepancy in an image. Generally, the higher the homogeneity is, the more similar the neighboring pixels in an image are.) For any loop-skipping level beyond L0, a correlation coefficient larger than or close to 0.7 is observed, which implies the energy saved and the homogeneity are positively and highly correlated. Therefore, we argue that the proposed loop-skipping technique is an effective approximation

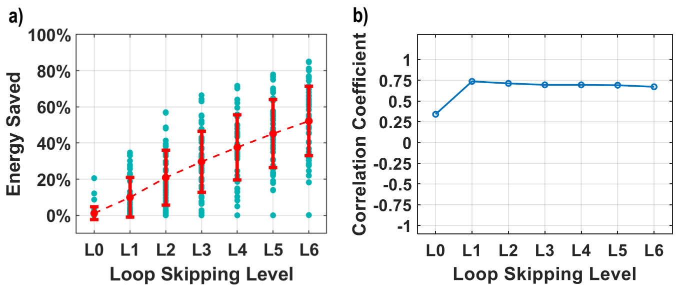

Fig. 20. Effect of different loop-skipping levels on a) energy saved and b) correlation coefficient between the energy saved and homogeneity.

scheme for signals exhibiting high spatial locality, like images, video, etc.

TABLE IV
STATISTICS OF FIG. 20: ENERGY SAVED UNDER DIFFERENT LOOP SKIPPING LEVELS.

Loop-Skipping Level	Energy Saved			
	mean	std	max	min
L_0	1.14 %	3.52 %	1.14 %	0
L_1	9.9 %	10.96 %	34.72 %	0
L_2	20.84 %	15.14 %	57.01 %	0
L_3	29.64 %	16.91 %	66.41 %	0
L_4	37.57 %	18.12 %	71.66 %	0
L_5	45.17 %	18.82 %	77.88 %	0
L_6	52.14 %	19.23 %	85.06 %	0.12 %

The trade-off of the energy saved by loop skipping versus the quality degradation of reconstructed images is outlined in Fig. 21. It plots the relation between relative energy and image quality degradation for (a) SAD and (b) SSIM. The result shows that 30% energy can be saved with roughly 2% SAD and 10% SSIM degradation at loop-skipping level L_3.

In the end, subjective analysis of loop perforation is shown in Fig. 22, which displays the reconstructed images of Baboon, Pirate, and Boat for different loop skipping levels.

D. DCT

Several multiplier-less DCT architectures have been proposed these years. For example, [27] proposed an efficient 1D-DCT structure that approximated the coefficients of the DCT transform matrix $\mathbf{T}$ as the power of 2 and exploited adjacent pixel correlation to reduce hardware complexity. Similarly, the idea of approximating $\mathbf{T}$'s coefficients as the power of 2 was applied in [28] to realize an approximate integer DCT scheme for HEVC. Other work such as [29], utilized approximate

TABLE V
COMPARISON WITH EXISTING MULTIPLIER-LESS DCT DESIGNS.

Item	This work	23'ASICON [27]	21'VLSI-SoC [28]	18'ISCAS [29]	15'ICVLSI [30]	15'ISVLSI [31]
Approximate Technique	CSD and CSE	1. **T** coefficients approx. 2. Optimized DCT by using adjacent pixel correlation	Approx. Integer DCT	1. **T** coefficients approx. 2. Approx. adders	An approx. DCT for only 25 coefficients	CSD and CSE
Technology	65 nm	28 nm	45 nm	180 nm	45 nm	90 nm
Area (um^2)	$6,500$	$12,687$	$3,217$	$876k$	$3,706$	$27,870$
Power (mW)	1.64	19.86	0.77	10.34	0.45	2.3
Energy (pJ)	16.4	198	14.6	77.9	29.2	23

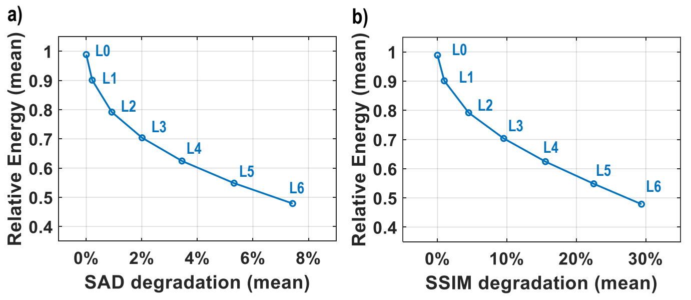

Fig. 21. Effect of different loop-skipping levels: a) relative energy vs. SAD degradation and b) relative energy vs. SSIM degradation.

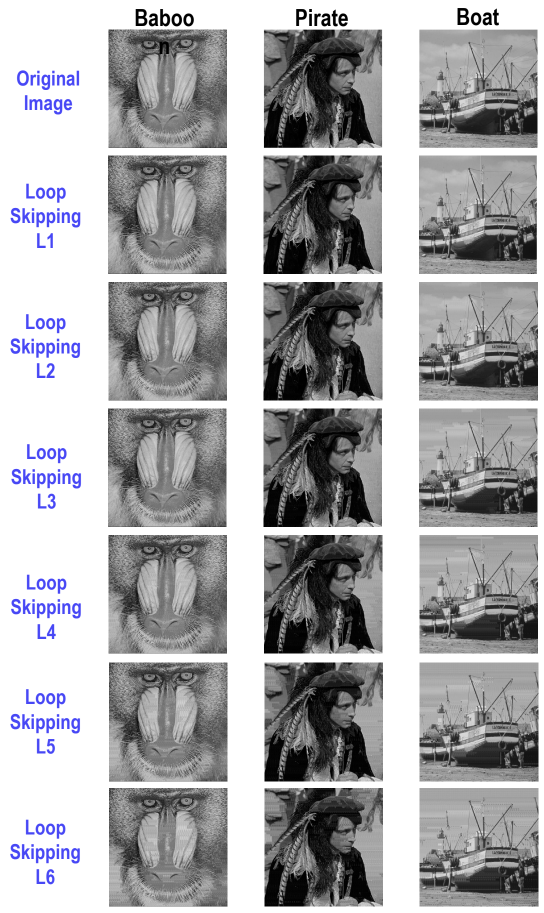

Fig. 22. Effect of loop skipping: reconstructed images for different loop skipping levels.

adders to build an energy-efficient DCT. [30] proposed a 25-coefficient DCT architecture by exploiting pixels' correlation and relative significance of DCT coefficients.

We compare the hardware simulation results of this work's DCT scheme, which uses the techniques of CSD (Canonic Signed Digit) and CSE (Common Subexpression Elimination) in [18], with other existing 8-point multiplier-less DCT works in Table. V. The 1D-DCT structure utilized in this paper is described in Verilog and synthesized by Synopsys Design Compiler with TSMC 65nm process. The circuit's power performance is measured by Cadence Spectre Simulator, where 256 8-by-1 column vectors of random pixels are sent as inputs to the circuit, and the average power consumption is measured under 1V supply and 100MHz clock. The final result shows that our 1D-DCT structure consumes 16.4pJ, which is better than most of the existing works.

E. Results from final architecture: quality, area & energy

The final JPEG compression circuit is synthesized using the Synopsys Design Compiler tool and mapped to TSMC 65nm library. The synthesis result shows that the final architecture (with the proposed bit-shifting-based quantization block and the loop skipping function) occupies a cell area of $109,470$ um^2, which is 28% less than the baseline (with conventional division-based quantization block and without the loop skipping function) design ($151,617$ um^2).

The area reduction comes mainly from improving the quantization step through the bit shift operators-based division, even though, at the same time, bit-truncation and loop skipping induce some area overhead. The baseline quantization blocks, comprising synthesis tool-generated dividers, consume 47% ($71,285$ um^2) of the entire area; however, the optimized quantization takes only 15% ($11,013$ um^2) of the total area. This area saving, $71,285 - 11,013 = 60,272$ um^2, outweighs the additional area resulting from the logic and registers used for bit-truncation and loop skipping, which takes $(109,470 - 11,013) - (151,617 - 71,285) = 98,457 - 80,332 = 18,125$ um^2. Our proposed methods combined, therefore, give an area saving of $42,147$ um^2. Fig. 23(a) shows the area comparison of the baseline and proposed design, where the red bars represent the area overhead of the quantization blocks and the blue bars represent the non-quantization part.

According to the simulation of Cadence Spectre Simulator, at TT corner, 25°C, with a supply of 1V, the baseline design (using bit truncation level 1 and loop skipping level 2) consumes an average current of 6.75 mA at 100MHz at the expense of 2% SAD degradation. On the other hand, the proposed architecture consumes an average current of 4.35 mA under the same simulation condition. Therefore, 36% energy is saved from our proposed approach, as shown in Fig. 23(b). The proposed architecture equivalently dissipates a power of 15uW in the DCT and quantization stages to generate a throughput of 480p colored image @ 6fps. This is 10× better than the analog solution [32] (which utilizes passive elements, i.e., switch capacitors to save power) and 6× better than current state-of-the-art [33] (which operates at near-threshold to reduce power).

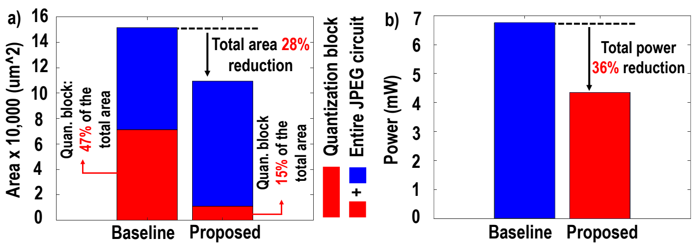

Fig. 23. Comparison between the baseline and proposed design: a) Area breakdown. b) Power consumption.

VI. Discussion

This work focuses on accelerating the DCT and quantization steps in JPEG compression. However, another widely used image compression scheme, JPEG2000, merits discussion due to its superior compression efficiency and image quality.

1) JPEG vs. JPEG2000 : Although the transform step in JPEG2000, which uses discrete wavelet transform (DWT), is simpler than the discrete cosine transform (DCT) in JPEG, the overall structure of JPEG2000 is more complex [34] [35] [36] [37] [38] [39]. This complexity arises from more refined quantization steps involving floating-point division and advanced coding schemes like Embedded Block Coding with Optimized Truncation (EBCOT) [40]. Consequently, JPEG2000 is not well-suited for applications in edge devices.

2) DCT vs. DWT: The 2D-DCT is more complex than the 2D-DWT for the transformation step. Nonetheless, the multiplier-less approach adopted in this design significantly reduces the hardware implementation cost, making it competitive with the 2D-DWT in terms of hardware resources. The transform block in the proposed design performs the 2D-DCT using only shift, addition, and subtraction operations—no multipliers are used. This approach is highly similar to the implementation of FIR-based 2D-DWT.

3) Possible future work for approximate JPEG2000 : The approximate techniques proposed in this paper—multiplier-less transformation, approximate quantization, bit truncation, and loop perforation—can be applied not only to JPEG but also to JPEG2000. For instance, 2D-DWT can be implemented using only shift, addition, and subtraction operations, as it is essentially an FIR filter. Quantization in JPEG2000 can also be implemented through shifting logic. Additionally, since JPEG2000 compresses images in small tiles, the concept of loop skipping can be utilized. If two neighboring tiles are sufficiently similar, the compression unit can be disabled, and the results from the last processed tile can be reused. Our future work can include detailed approximated hardware analysis and implementation and related encoded/decoded image analysis for JPEG2000.

VII. Conclusion

This work demonstrates a synthesizable multiplier-less JPEG accelerator equipped with approximations both in software and RTL in the form of modified quantization block, precision scaling, and loop perforation, trading off the quality of the image with energy & area reduction. With a gradient descent-based heuristic, the accelerator's performance can be tuned to maximize energy savings while meeting the image quality constraints. The proposed architecture with the combined approximation strategies achieves 36% reduction in energy consumption at the expense of 2% SAD quality degradation in the image, which lies within acceptable limits for any image processing applications. Moreover, it consumes 15uW at the DCT and quantization stages to compress a colored 480p image at 6fps, which is 10x better than the previous literature.

References

[1] Zhu et al. Lossless Image Compression Algorithm Based on Long Short-term Memory Neural Network. In *2020 5th ICCIA*, pages 82–88, 2020.

[2] Lu Liang et al. Study on JPEG2000 Optimized Compression Algorithm for Remote Sensing Image. In *NSWCTC*, volume 2, pages 771–775, 2009.

[3] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.

[4] C. Christopoulos, A. Skodras, and T. Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, 2000.

[5] Farzad Ebrahimi, Matthieu Chamik, and Stefan Winkler. Jpeg vs. jpeg 2000: an objective comparison of image encoding quality. In *Applications of Digital Image Processing XXVII*, volume 5558, pages 300–308. SPIE, 2004.

[6] Elizabeth Allen, Sophie Triantaphillidou, and R Jacobson. Image quality comparison between jpeg and jpeg2000. i. psychophysical investigation. *Journal of Imaging Science and Technology*, 51(3):248–258, 2007.

[7] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. Low-complexity 8× 8 transform for image compression. *Electronics Letters*, 44(21):1249–1250, 2008.

[8] Saad Bouguezel, M. Omair Ahmad, and M. N. S. Swamy. A fast 8×8 transform for image compression. In *2009 International Conference on Microelectronics - ICM*, pages 74–77, 2009.

[9] Saad Bouguezel, M. Omair Ahmad, and M.N.S. Swamy. A low-complexity parametric transform for image compression. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2145–2148, 2011.

[10] Nabila Brahimi, Toufik Bouden, Tahar Brahimi, and Larbi Boubchir. A novel and efficient 8-point dct approximation for image compression. *Multimedia Tools and Applications*, 79, 2020.

[11] Snigdha et al. Optimal design of JPEG hardware under the approximate computing paradigm. In *2016 53nd ACM/EDAC/IEEE DAC*, pages 1–6, 2016.

[12] Gupta et al. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE TCAD*, 32(1):124–137, 2013.

[13] Park et al. Dynamic Bit-Width Adaptation in DCT: An Approach to Trade off Image Quality and Computation Energy. *IEEE TVLSI*, 18(5):787–793, 2010.

[14] Haider A.F. Almurib, Thulasiraman Nandha Kumar, and Fabrizio Lombardi. Approximate dct image compression using inexact computing. *IEEE Transactions on Computers*, 67(2):149–159, 2018.

[15] Imani et al. CADE: Configurable Approximate Divider for Energy Efficiency. In *2019 DATE*, pages 586–589, 2019.
[16] G.K. Wallace. The jpeg still picture compression standard. *IEEE T. Consumer Electronics*, 38(1):xviii–xxxiv, 1992.
[17] Wen-Hsiung Chen, C. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, 25(9):1004–1009, 1977.
[18] Byoung-Il Kim and Sotirios G. Ziavras. Low-power multiplierless dct for image/video coders. In *2009 IEEE 13th International Symposium on Consumer Electronics*, pages 133–136, 2009.
[19] Jackson et al. SAADI-EC: A Quality-Configurable Approximate Divider for Energy Efficiency. *IEEE TVLSI*, 27(11):2680–2692, 2019.
[20] Hashemi et al. A low-power dynamic divider for approximate applications. In *2016 53nd ACM/EDAC/IEEE DAC*, pages 1–6, 2016.
[21] Vahdat et al. TruncApp: A truncation-based approximate divider for energy efficient DSP applications. In *DATE, 2017*, pages 1635–1638, 2017.
[22] University of Granada Test Images. https://ccia.ugr.es/cvg/index2.php. [Online, accessed 31 Jan. 2024].
[23] Salvatore Barone, Marcello Traiola, Mario Barbareschi, and Alberto Bosio. Multi-objective application-driven approximate design method. *IEEE Access*, 9:86975–86993, 2021.
[24] Mario Barbareschi, Salvatore Barone, Alberto Bosio, Jie Han, and Marcello Traiola. A genetic-algorithm-based approach to the design of dct hardware accelerators. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(3):1–25, 2022.
[25] Borko Furht, editor. *Discrete Cosine Transform (DCT)*, pages 186–188. Springer US, Boston, MA, 2008.
[26] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973.
[27] Xu Wang, Ke Chen, Chenghua Wang, and Weiqiang Liu. An energy-efficient approximate dct design for image processing. In *2023 IEEE 15th International Conference on ASIC (ASICON)*, pages 1–4. IEEE, 2023.
[28] S Skandha Deepsita, Kuchipudi Divya, and S Noor Mahammad. Energy efficient and multiplierless approximate integer dct implementation for hevc. In *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2021.
[29] Yan Xing, Ziji Zhang, Yiduan Qian, Qiang Li, and Yajuan He. An energy-efficient approximate dct for wireless capsule endoscopy application. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2018.
[30] Vikas Kaushal, Bharat Garg, Ankur Jaiswal, and G.K. Sharma. Energy aware computation driven approximate dct architecture for image processing. In *2015 28th International Conference on VLSI Design*, pages 357–362, 2015.
[31] Anand D Darji and Raviraj P Makwana. High-performance multiplierless dct architecture for hevc. In *2015 19th International Symposium on VLSI Design and Test*, pages 1–5. IEEE, 2015.
[32] K Gaurav Kumar, Gourab Barik, Baibhab Chatterjee, Sumon Bose, Shovan Maity, and Shreyas Sen. A 65 nm 2.02 mw 50 mbps direct analog to mjpeg converter for video sensor nodes using low-noise switched capacitor mac-quantizer with automatic calibration and sparsity-aware adc. In *2023 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–2, 2023.
[33] Nele Reynders et al. 27.3 a 210mv 5MHz variation-resilient near-threshold JPEG encoder in 40nm CMOS. In *2014 ISSCC*, pages 456–457, 2014.
[34] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen, and Liang-Gee Chen. Analysis and architecture design of block-coding engine for ebcot in jpeg 2000. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(3):219–230, 2003.
[35] Michael D Adams. The jpeg-2000 still image compression standard. *ISO/IEC JTC 1/SC 29/WG 1 N 2412, Dec.*, 2002.
[36] Liang-Gee Chen, Chung-Jr Lian, Kuanfu Chen, and Hong-Hui Chen. Analysis and architecture design of jpeg2000. In *ICME*. Citeseer, 2001.
[37] Yicong Meng, Leibo Liu, Li Zhang, and Zhihua Wang. Design methodology of low power jpeg2000 codec exploiting dual voltage scaling. In *2005 6th International Conference on ASIC*, volume 1, pages 183–186, 2005.
[38] ALIM REZA. Jpeg2000 hardware implementation procedures and issues. *WSEAS Transactions on Circuits and Systems*, 12, 2013.
[39] Leibo Liu, Ning Chen, Hongying Meng, Li Zhang, Zhihua Wang, and Hongyi Chen. A vlsi architecture of jpeg2000 encoder. *IEEE Journal of Solid-State Circuits*, 39(11):2032–2040, 2004.
[40] D. Taubman. High performance scalable image compression with ebcot. *IEEE Transactions on Image Processing*, 9(7):1158–1170, 2000.

Ming-Che Li (Graduate Student Member, IEEE) was born in Yilan, Taiwan, in 1999. He received the B.S. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2021. He is currently pursuing a Ph.D. degree in electrical and computer engineering at Purdue University, West Lafayette, IN, USA.

His current research interests include hardware security, approximate computing in image & video compression, and stochastic computing.

Archisman Ghosh (Student Member, IEEE) received his B.E. degree in Electronics and Telecommunication Engineering from Jadavpur University, India, in 2017, and he is currently pursuing a Ph.D. at Purdue University, where he was a recipient of prestigious ECE Meissner fellowship (2019-20) as an incoming graduate student. He is currently a Bilsland Dissertation fellow at Purdue University.

His research interests include digital SoC design and hardware security. Prior to his Ph.D., Mr. Ghosh worked in Samsung Semiconductor India R&D for 2 years. He has interned with Intel Labs, Oregon. He is one of the recipients of the prestigious IEEE SSCS Pre-doctoral Achievement Award 2022.

Shreyas Sen (Senior Member, IEEE) is an Elmore Associate Professor of ECE & BME, Purdue University. His current research interests span mixed-signal circuits/systems and electromagnetics for the Internet of Bodies (IoB) and Hardware Security. He has authored/co-authored 3 book chapters, over 200 journal and conference paper and has 25 patents granted/pending. Dr. Sen serves as the Director of the Center for Internet of Bodies (C-IoB) at Purdue. Dr. Sen is the inventor of the Electro-Quasistatic Human Body Communication (EQS-HBC), or Body as a Wire technology, for which, he is the recipient of the MIT Technology Review top-10 Indian Inventor Worldwide under 35 (MIT TR35 India) Award in 2018 and Georgia Tech 40 Under 40 Award in 2022. To commercialize this invention Dr. Sen founded Ixana and serves as the Chairman and CTO and led Ixana to awards such as 2x CES Innovation Award 2024, EE Times Silicon 100, Indiana Startup of the Year Mira Award 2023. His work has been covered by 250+ news releases worldwide, invited appearance on TEDx Indianapolis, NASDAQ live Trade Talks at CES 2023, Indian National Television CNBC TV18 Young Turks Program, NPR subsidiary Lakeshore Public Radio and the CyberWire podcast. Dr. Sen is a recipient of the NSF CAREER Award 2020, AFOSR Young Investigator Award 2016, NSF CISE CRII Award 2017, Intel Outstanding Researcher Award 2020, Google Faculty Research Award 2017, Purdue CoE Early Career Research Award 2021, Intel Labs Quality Award 2012 for industrywide impact on USB-C type, Intel Ph.D. Fellowship 2010, IEEE Microwave Fellowship 2008, GSRC Margarida Jacome Best Research Award 2007, and nine best paper awards including IEEE CICC 2019, 2021 and in IEEE HOST 2017-2020, for four consecutive years. Dr. Sen's work was chosen as one of the top-10 papers in the Hardware Security field (TopPicks 2019). He serves/has served as an Associate Editor for IEEE Solid-State Circuits Letters (SSC-L), Nature Scientific Reports, Frontiers in Electronics, IEEE Design & Test, Executive Committee member of IEEE Central Indiana Section and Technical Program Committee member of TPC member of ISSCC, CICC, DAC, CCS, IMS, DATE, ISLPED, ICCAD, ITC, and VLSI Design. Dr. Sen is a Senior Member of IEEE.