Compressive Acquisition CMOS Image Sensor: From the Algorithm to Hardware Implementation

Milin Zhang, Student Member, IEEE, and Amine Bermak, Senior Member, IEEE

Abstract—In this paper, a new design paradigm referred to as compressive acquisition CMOS image sensors is introduced. The idea consists of compressing the data within each pixel prior to storage, and hence, reducing the size of the memory required for digital pixel sensor. The proposed compression algorithm uses a block-based differential coding scheme in which differential values are captured and quantized online. A time-domain encoding scheme is used in our CMOS image sensor in which the brightest pixel within each block fires first and is selected as the reference pixel. The differential values between subsequent pixels and the reference within each block are calculated and quantized, using a reduced number of bits as their dynamic range is compressed. The proposed scheme enabled reduced error accumulation as full precision is used at the start of each block, while also enabling reduced memory requirement, and hence, enabling significant silicon area saving. A mathematical model is derived to analyze the performance of the algorithm. Experimental results on a field-programmable gate-array (FPGA) platform illustrate that the proposed algorithm enablers more than 50% memory saving at a peak signal-to-noise ratio level of 30 dB with 1.5 bit per pixel.

Index Terms—Compressive acquisition, digital pixel sensors (DPSs), differential pulse code modulation (DPCM), online block-based compression algorithm.

I. INTRODUCTION

T HE wide spread of video cameras, internet video, mobile imaging, and biomedical microsystem applications has greatly increased the technology demand for low-power sensing devices as well as extensive image processing capabilities [1], [2]. Increased resolution and frame rate have also placed increased burden in terms of data to be processed and stored. Compression is therefore becoming increasingly important for video and still imaging devices. In the last decades, various image compression algorithms were proposed and established as standards, such as the joint photographic experts group (JPEG) standards [3], [4], the discrete cosine transform (DCT)-based coding [5]–[7], the wavelet-based coding [8]–[11], etc. However, the complexity of these compression algorithms is very high. Implementing such algorithms into hardware is typically costly and consumes significant power [12]–[14]. Recent hardware friendly compression algorithms [15]–[18] have emerged and are being deployed in a number of emerging new technologies such as camera pill [19] and sensor network applications [20]. Most of these applications require the use of extensive on-chip processing, which is only possible using CMOS technology. Indeed, CMOS image sensors are receiving more and more attention over the conventional charge-coupled device (CCD) image sensor [21]. Among the various architectures of CMOS image sensors, the digital pixel sensor (DPS) [22]–[24] provides digital outputs that can be directly used for pixel-level processing [25]. Unfortunately, the large pixel area required for pixel-level storage and the low fill-factor performance have seriously prevented DPS from being the technology of choice despite its tremendous potential. Indeed, in a DPS sensor, even more than half of the area is occupied by the pixel-level memory [30], [31] making it a real burden for any industrial application (low yield due to a large pixel size). Recent research work explored moving the memory out of the pixel [33], [34] but this results in slower image sensor as the whole concept of parallel image capture inherent in DPS structure is given up.

In this paper, we propose to overcome this high-memory requirement issue in DPS by shifting the design paradigm from the traditional concept of capture ↔ store ↔ compress to a new design paradigm namely: capture ↔ compress ↔ store. The idea consists of compressing the data within each pixel before storing making it possible to use fewer bits at the pixel level, and hence, reducing the size of the memory required for DPS sensors. The advantage of the proposed concept are three folds: 1) reduced silicon area required for the DPS; 2) increased fill-factor; and 3) compression processing integrated within the pixel array, enabling the concept of parallel processing. A mathematical model is derived in order to optimize the quantization step of the quantizer so as to achieve maximum performance regardless of the block size. Results illustrate that the proposed algorithm can result in more than 50% on-chip memory saving at a peak signal-to-noise ratio (PSNR) level of about 30 dB.

This paper is organized as follows. Section II introduces the concept behind the proposed design paradigm and the associated compression algorithm. Section III proposes the mathematical derivation for the optimum quantization step as well as the simulation results. Section IV describes the hardware architecture of the proposed algorithm, while Section V reports the experimental test results using a field-programmable gate array (FPGA) platform interfaced to an image sensor. Analysis of performance based on layout design is also shown in this section to support the theoretical claims. Section VI concludes this paper.

II. ALGORITHMIC DESCRIPTION

A. Compressive Acquisition—A New Design Paradigm

In a traditional image sensor architecture, the raw pixel values are captured using a sensor array converting illumination...
intensity into a current or a voltage signal. Analog signals are then converted to digital data using analog-to-digital converters. The digital pixel data are stored in on-chip memories for further digital processing. Data compression is increasingly becoming one of the most important processing operations due to increased resolution and frame rate in today’s imaging systems. Compression algorithms typically require extensive processing power and large amount of memories. Typically, storage is performed off-array in order to maintain competitive pixel size and fill-factor. There are, however, a number of attempts looking at the possibility of embedding memory into the pixel, leading to the so-called DPS. Despite the potential of DPS (high parallelism due to the parallel structure of ADCs), it never made a true industrial impact due to the large pixel size leading to limited resolution, poor fill-factor, low yield, and hence high cost. Typically, in a DPS structure, the illumination level is converted to a voltage that is then quantized and digitized before being stored in a pixel-level memory. The data will then undergo a compression procedure to reduce the amount of data to be transmitted. A typical signal flow is therefore described by the three basic steps namely: capture \(\rightarrow\) store \(\rightarrow\) compress. In this paper, we propose to address the issue of high-storage requirement in DPS by compressing the data online during the image capture procedure prior to the storage phase. Storing the compressed data will lead to reduced memory requirement at the pixel level, and hence, smaller pixel size and improved DPS fill-factor are achieved. This approach will shift the design paradigm from the conventional: capture \(\rightarrow\) store \(\rightarrow\) compress to the newly proposed: Capture \(\rightarrow\) compress \(\rightarrow\) store, as illustrated in Fig. 1.

The basic idea behind a compressive-acquisition CMOS image sensor is to compress data while performing image acquisition. One may think about this process as combining image capture with compression, enabling the use of fewer memory cells to store an effectively higher pixel-intensity resolution. It is obviously a tradeoff, as performing compression prior to storage involves incorporating a processing unit at the pixel level, as shown in Fig. 1, which may further complicate the pixel design. The key question affecting the success of this approach is whether it is possible to simplify the compression processing to enable a powerful and yet simple pixel design.

Another key consideration is related to the fact that compression is typically performed by removing spatial and temporal redundancies within the array. If the compression processing unit is embedded at the pixel level, how are these redundancies ever visible to the pixel-based processing unit? The compressive acquisition CMOS image sensor answers these questions by acquiring the illumination signals of a group of pixels at the same time instead of looking at a single pixel and using full-resolution storage for every pixel. The grouping of pixels can be done taking into consideration spatial and temporal dimensions. In this paper, we propose a block-based compression algorithm, which can be implemented at the pixel-level with very limited hardware complexity, and hence, limited overhead in terms of silicon area.

### B. Block-Based Compression Algorithm

It is well known that natural images present a certain degree of spatial and temporal redundancy, which compression algorithms attempt to remove. To understand how this works in our compressive acquisition, let us assume for simplicity that we are interested in removing spatial redundancy using differential coding. The theoretical foundations behind such a scheme are explained by the fact that the distribution of the differential values between pixels and their predictions is more regular and can be broadly modeled as a Gaussian-like distribution. Differential coding presents the advantage of reduced dynamic range, enabling the possibility of using a reduced number of bits. For practicality, let us assume that we would like to simply quantize raw data from an image sensor, as illustrated in Fig. 2. The pixel value is first subtracted from its prediction value (which can simply be the neighboring pixel) and the result is then quantized and transmitted. On the receiver side, the pixel value is reconstructed by performing the reverse operation. One obvious drawback for such a scheme is the accumulation of errors due to sharp edges in the captured scene. This can be illustrated in the reconstructed Lenna picture, as illustrated in Fig. 2. Fig. 3 shows in more details how the error between the reconstructed pixel value and the raw pixel value accumulates while scanning data out. This obviously presents a serious drawback in differential coding.

In order to overcome this error accumulation problem inherent in differential coding, we propose to regularly use “reference pixels” in which full precision is used to encode the absolute value instead of the differential data. The choice in terms of the number of reference pixels is a tradeoff between image quality and compression ratio. In our compression scheme, we propose to decompose the image into blocks. Within each block,
Fig. 3. Absolute error value between the reconstructed image and the raw image. The plot illustrates a single row data.

A single pixel is used as a reference pixel, in which full-precision of 8-bit is used. The differential values between subsequent pixel values within the block and the reference pixel value are calculated and quantized, using a reduced number of bits (see Fig. 4) as their dynamic range is compressed. As a consequence, the memory requirement for a given block will be heavily reduced, resulting in a more efficient pixel array implementation. The reference pixel is selected to be the brightest pixel within the block, so as to enable the differential values to be always positive. This further enables a compression of the dynamic range by a factor of two as compared to conventional differential pulse code modulation (DPCM), in which differential values can be positive or negative. In addition, since the differential values are always obtained with respect to a single reference pixel, accumulation of error is completely avoided.

The differential coding scheme explained earlier is only a first step in compressive-acquisition processing. Since a block-based partition of the array is performed in this first step, further spatial redundancy removal can be achieved using further decomposition, such as quadrant tree decomposition. Indeed, a tree is constructed hierarchically whereby multiple hierarchical layers of the tree, corresponding each to a block, are formed while reading the block-based differential values. Fig. 5 describes the addressing strategy for quadrant tree decomposition. First, the block is divided into four subblocks. Each subblock will be associated with a leaf within the primary tree. Each subblock is further divided into four subblocks if the block is larger than $2 \times 2$. A tree using $\log_2 (K \times K)$ layers is therefore formed as illustrated in Fig. 5. The leaves of the tree correspond to the comparison results between the differential values $\delta(i,j)$ and a threshold $X_{th}$. If $\delta(i,j)$ is smaller than $X_{th}$, a second layer flag is set to one, otherwise, it is set to zero. If the leave flags attached to the same parent node are all equal to 1, then a higher layer flag is set to one which means that all differential values within that specific block can be compressed and represented by the value of the brightest pixel $M(i)$. The flag is set to one or zero indicating whether the differential values of the corresponding block can or cannot be compressed. The tree is therefore built in one iteration during the scanning procedure of the pixel array. The proposed scanning scheme attempts to remove spatial redundancy by compressing similar differential values belonging to the same block and representing them by a single value.
In summary, there are two steps involved in the proposed compression algorithm:

1) block-based division of the raw image and quantization of the differential values between the brightest pixel and subsequent pixels within the same block;

2) \( \log_2(K \times K) \) layers flag tree generation enabling spacial redundancy removal.

It should be noted that there is a fundamental difference between the proposed algorithm and other standard compression algorithms. The central idea behind compressive acquisition is to remove image redundancy prior to storage. In the proposed scheme, image capture needs to go hand-in-hand with compression. The memory in this proposed scheme stores the compressed image right after capture. At the receiver side, decompression is performed in a very simple way as it involves reconstructing raw data from differential values and a reference pixel value.

III. MATHEMATICAL MODELING AND SIMULATION RESULTS

A. Performance Optimization

As explained in the previous section, the differential values within each block are quantized using limited number of bits (typically, 2–3 bits). With this limited precision, the choice of the quantization step becomes very critical. In this section, we will develop a mathematical model that enables to set the quantization step length \( \delta \) to be optimal. We shall derive the best quantization step length \( \delta \), which optimizes the PSNR, or in other words, minimizes the quantization error.

Using the Gaussian-like distribution, we can model \( P_n \) as follows:

\[
P(n) = f(x = n), \quad (n = 0, 1, 2, \ldots, D_{\text{max}})
\]

with

\[
f(x) = \frac{1}{\sqrt{2\pi} \sigma} e^{-x^2/2\sigma^2}, \quad P(0) = \frac{1}{\sqrt{2\pi} \sigma}
\]

where \( f(x) \) is the expression of the continuous Gaussian distribution function with the variance \( \sigma \). \( D(i, j) \in [0, D_{\text{max}}] \), where \( D_{\text{max}} \) is the maximum value of the differential value. Typically, \( D(i, j) \) is an 8-bit data, with a maximum value \( D_{\text{max}} \) equals to 255. The fitting error between the proposed distribution model as expressed in (1) and the real distribution is defined as \( \varepsilon_{\text{fit}} \), which can be expressed by the mse

\[
\varepsilon_{\text{fit}} = \sum_{i=0}^{D_{\text{max}}} (P(i) - P_i)^2.
\]

A best variance \( \sigma_{\text{best}} \) of the distribution model expressed in (1) is defined as the value of \( \sigma \), which minimizes the fitting error \( \varepsilon_{\text{fit}} \) to \( \varepsilon_{(\varepsilon_{\text{fit}})_{\text{min}}} \). There are two factors affecting the value of \( \sigma_{\text{best}} \). One is the size of the block \( K \), the other is the raw image data itself. Fig. 7 shows the integral value of \( \sigma_{\text{best}} \) for different block sizes and different sample images. One can note that for a fixed block size, the average \( \sigma_{\text{best}} \) value can be used for all sample images without affecting the performance. In other words, for a fixed block size, it is possible to deduce a close to optimum value of \( \sigma_{\text{best}} \) that can fit different sample images.

The differential value \( D(i, j) \) is quantized using an \( N \)-bit fixed step quantizer. For a fixed variance \( \sigma \), now we shall derive the best quantization step length \( \delta \), which optimizes the PSNR, or in other words, minimizes the quantization error.
The quantized differential value \( \hat{D}(i, j) \) can be expressed as follows:

\[
\hat{D}(i, j) = \begin{cases} 
0 & \text{when } D(i, j) \in [0, \delta[ \\
1 & \text{when } D(i, j) \in [\delta, 2\delta[ \\
& \vdots \\
(2^{N} - 2) & \text{when } D(i, j) \in \left[(2^{N} - 2)\delta, (2^{N} - 1)\delta[ \\
(2^{N} - 1) & \text{when } D(i, j) \in \left[(2^{N} - 1)\delta, D_{\max}\right]
\end{cases}
\] (3)

where \( \delta \) stands for the step length of the quantizer. The quantization error is defined as

\[
\varepsilon = \sum_{n=0}^{2^{N} - 1} \sum_{m=0}^{(m+1)\delta - 1} (n - m\delta)^{2} P(n) + \sum_{n=1}^{D_{\max}} (n - (2^{N} - 1)\delta)^{2} P(n).
\] (4)

A best step length \( \delta_{\text{best}} \) is defined as the value of \( \delta \) which minimizes the quantization error \( \varepsilon \) to \( \varepsilon_{\text{min}} \). \( \delta_{\text{best}} \) can be found by solving the following expression:

\[
\frac{\partial \varepsilon}{\partial \delta} = 0
\] (5)

where (5) can be approximated as

\[
\frac{\partial \varepsilon}{\partial \delta} = 0 \iff \Phi(\delta - 1) = 0
\] (6)

with

\[
\Phi(x) = -x^{4} + \left( \frac{4\sqrt{2\pi}\sigma}{3} - 2 \right) x^{3} + 2\sqrt{2\pi}\sigma x^{2} - 4\sigma^{4}.
\]

Detailed derivation of (6) can be found in Appendix A. The solution of (6) provides a relationship between the variance \( \sigma \) of the distribution model (1) and the best step length \( \delta_{\text{best}} \), as illustrated in Fig. 8. As expression (6) is a degree 4 polynomial function, there is no direct solution of \( \delta_{\text{best}} \), and hence, an iterative solution is derived. The final results are illustrated in Fig. 8.

The quantization error is defined as the value of

\[
\varepsilon = \sum_{n=0}^{2^{N} - 1} \sum_{m=0}^{(m+1)\delta - 1} (n - m\delta)^{2} P(n) + \sum_{n=1}^{D_{\max}} (n - (2^{N} - 1)\delta)^{2} P(n).
\]

A best step length \( \delta_{\text{best}} \) is defined as the value of \( \delta \) which minimizes the quantization error \( \varepsilon \) to \( \varepsilon_{\text{min}} \). \( \delta_{\text{best}} \) can be found by solving the following expression:

\[
\frac{\partial \varepsilon}{\partial \delta} = 0
\]

where (5) can be approximated as

\[
\frac{\partial \varepsilon}{\partial \delta} = 0 \iff \Phi(\delta - 1) = 0
\]

with

\[
\Phi(x) = -x^{4} + \left( \frac{4\sqrt{2\pi}\sigma}{3} - 2 \right) x^{3} + 2\sqrt{2\pi}\sigma x^{2} - 4\sigma^{4}.
\]

Detailed derivation of (6) can be found in Appendix A. The solution of (6) provides a relationship between the variance \( \sigma \) of the distribution model (1) and the best step length \( \delta_{\text{best}} \), as illustrated in Fig. 8. As expression (6) is a degree 4 polynomial function, there is no direct solution of \( \delta_{\text{best}} \), and hence, an iterative solution is derived. The final results are illustrated in Fig. 8.

The compression ratio of the first step compression \( CR_{1} \) is expressed in

\[
CR_{1} = \frac{8 \times K \times K}{N \times K \times K + 8} = \frac{8}{N + \frac{8}{K^{2}}} \to \frac{8}{N}
\]

where \( K \) stands for the block size and \( N \) stands for the resolution of the quantizer. The compression ratio approaches \( 8/N \) when \( K \) is large. The compression ratio of the second step compression \( CR_{2} \) depends on the spatial redundancy of the sample image that can be obtained only through simulation.

### B. Simulation Results

System level simulation is carried out on sample images. The best step length of different block sizes is set based on the mathematical derivation (6). The nearest integer solution of expression (6) is utilized as the best step length during simulation. Table I shows the compression results for some sample images with different block sizes \( K \) and different quantizer resolutions \( N \), when the threshold value \( X_{\text{th}} = 1 \). \( CR_{1} \) and \( CR_{2} \) stand for compression ratio of the first and second compression steps, respectively. The value of \( CR_{1} \) depends on the block size and resolution of the quantizer. It can be directly calculated using expression (7).

<table>
<thead>
<tr>
<th>Testing Samples</th>
<th>Block Size</th>
<th>( Q - \text{bit}^{1} )</th>
<th>PSNR</th>
<th>( CR_{1} )</th>
<th>( CR_{2} )</th>
<th>BPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane</td>
<td>3</td>
<td>28.86</td>
<td>2.77</td>
<td>1.65</td>
<td>1.74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>33.80</td>
<td>2.06</td>
<td>1.84</td>
<td>2.11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>27.52</td>
<td>3.20</td>
<td>1.86</td>
<td>1.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>31.94</td>
<td>2.29</td>
<td>2.02</td>
<td>1.73</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>27.24</td>
<td>3.45</td>
<td>1.98</td>
<td>1.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30.20</td>
<td>2.41</td>
<td>2.11</td>
<td>1.57</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>26.84</td>
<td>3.60</td>
<td>2.02</td>
<td>1.10</td>
<td></td>
</tr>
<tr>
<td>Bonn</td>
<td>3</td>
<td>30.97</td>
<td>2.06</td>
<td>1.66</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>24.11</td>
<td>3.20</td>
<td>1.06</td>
<td>2.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>29.11</td>
<td>2.29</td>
<td>1.07</td>
<td>3.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>24.54</td>
<td>3.45</td>
<td>1.07</td>
<td>2.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>27.24</td>
<td>2.41</td>
<td>1.09</td>
<td>3.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>23.35</td>
<td>3.60</td>
<td>1.07</td>
<td>2.07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>26.34</td>
<td>2.48</td>
<td>1.08</td>
<td>2.99</td>
<td></td>
</tr>
<tr>
<td>Lenna</td>
<td>3</td>
<td>30.80</td>
<td>2.77</td>
<td>1.60</td>
<td>1.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>28.96</td>
<td>3.20</td>
<td>1.80</td>
<td>1.39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>31.36</td>
<td>2.29</td>
<td>1.94</td>
<td>1.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>27.94</td>
<td>3.45</td>
<td>1.90</td>
<td>1.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>29.49</td>
<td>2.41</td>
<td>2.01</td>
<td>1.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>27.23</td>
<td>3.60</td>
<td>1.92</td>
<td>1.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>28.53</td>
<td>2.48</td>
<td>1.99</td>
<td>1.62</td>
<td></td>
</tr>
<tr>
<td>Peppers</td>
<td>3</td>
<td>29.36</td>
<td>2.77</td>
<td>1.66</td>
<td>1.74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>32.90</td>
<td>2.06</td>
<td>1.83</td>
<td>2.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>28.16</td>
<td>3.20</td>
<td>1.87</td>
<td>1.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>31.16</td>
<td>2.29</td>
<td>2.02</td>
<td>1.73</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>27.38</td>
<td>3.45</td>
<td>2.00</td>
<td>1.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>29.34</td>
<td>2.41</td>
<td>2.13</td>
<td>1.56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>26.75</td>
<td>3.60</td>
<td>1.98</td>
<td>1.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>28.34</td>
<td>2.48</td>
<td>2.08</td>
<td>1.55</td>
<td></td>
</tr>
</tbody>
</table>

\(^{1}\) Q-bit stands for quantizer’s resolution

The compression ratio of the second compression \( CR_{2} \) depends on the spatial redundancy of the sample image that can be obtained only through simulation.
TABLE II

<table>
<thead>
<tr>
<th>$X_{th}$</th>
<th>Samples</th>
<th>$K = 3$</th>
<th>$K = 4$</th>
<th>$K = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Airplane</td>
<td>33.80 / 2.11</td>
<td>31.93 / 1.73</td>
<td>30.20 / 1.57</td>
</tr>
<tr>
<td></td>
<td>Clock</td>
<td>33.59 / 1.92</td>
<td>32.42 / 1.54</td>
<td>31.05 / 1.39</td>
</tr>
<tr>
<td></td>
<td>Elaine</td>
<td>32.83 / 2.64</td>
<td>30.22 / 2.22</td>
<td>28.24 / 1.99</td>
</tr>
<tr>
<td></td>
<td>Lenna</td>
<td>33.66 / 2.20</td>
<td>31.36 / 1.80</td>
<td>29.49 / 1.65</td>
</tr>
<tr>
<td></td>
<td>Man</td>
<td>32.37 / 2.65</td>
<td>30.16 / 2.26</td>
<td>28.29 / 2.10</td>
</tr>
<tr>
<td></td>
<td>Peppers</td>
<td>32.90 / 2.12</td>
<td>31.16 / 1.73</td>
<td>29.34 / 1.56</td>
</tr>
<tr>
<td></td>
<td>Zelda</td>
<td>33.86 / 1.78</td>
<td>31.61 / 1.43</td>
<td>29.76 / 1.26</td>
</tr>
<tr>
<td>2</td>
<td>Airplane</td>
<td>28.18 / 1.67</td>
<td>26.11 / 1.31</td>
<td>24.39 / 1.15</td>
</tr>
<tr>
<td></td>
<td>Clock</td>
<td>27.50 / 1.63</td>
<td>25.51 / 1.27</td>
<td>23.89 / 1.12</td>
</tr>
<tr>
<td></td>
<td>Elaine</td>
<td>29.64 / 1.49</td>
<td>27.57 / 1.06</td>
<td>25.90 / 0.85</td>
</tr>
<tr>
<td></td>
<td>Lenna</td>
<td>28.63 / 1.60</td>
<td>26.50 / 1.22</td>
<td>24.84 / 1.05</td>
</tr>
<tr>
<td></td>
<td>Man</td>
<td>29.27 / 1.79</td>
<td>27.28 / 1.42</td>
<td>25.74 / 1.24</td>
</tr>
<tr>
<td></td>
<td>Peppers</td>
<td>28.55 / 1.49</td>
<td>26.66 / 1.13</td>
<td>25.02 / 0.98</td>
</tr>
<tr>
<td></td>
<td>Zelda</td>
<td>28.37 / 1.29</td>
<td>26.39 / 0.90</td>
<td>24.75 / 0.69</td>
</tr>
<tr>
<td>3</td>
<td>Airplane</td>
<td>27.94 / 1.46</td>
<td>25.90 / 1.11</td>
<td>24.20 / 0.93</td>
</tr>
<tr>
<td></td>
<td>Clock</td>
<td>27.35 / 1.50</td>
<td>26.35 / 1.12</td>
<td>23.73 / 0.97</td>
</tr>
<tr>
<td></td>
<td>Elaine</td>
<td>29.22 / 1.19</td>
<td>27.26 / 0.77</td>
<td>25.65 / 0.59</td>
</tr>
<tr>
<td></td>
<td>Lenna</td>
<td>28.30 / 1.36</td>
<td>26.21 / 0.97</td>
<td>24.53 / 0.79</td>
</tr>
<tr>
<td></td>
<td>Man</td>
<td>28.75 / 1.44</td>
<td>26.80 / 1.05</td>
<td>25.24 / 0.84</td>
</tr>
<tr>
<td></td>
<td>Peppers</td>
<td>28.32 / 1.31</td>
<td>26.43 / 0.93</td>
<td>25.01 / 0.77</td>
</tr>
<tr>
<td></td>
<td>Zelda</td>
<td>28.18 / 1.14</td>
<td>26.20 / 0.73</td>
<td>24.57 / 0.52</td>
</tr>
</tbody>
</table>

1 all the numbers shown in this table are in the form of $PSNR(dB)/BPP(bit)$

and threshold values. The table illustrates an expected higher compression ratio and poorer image quality for larger threshold and block size. In order to deduce some design guidelines with respect to the optimum block size, we performed exhaustive simulation on various sample images, as illustrated in Fig. 9. The figure illustrates the PSNR to BPP ratio as a function of the block size and threshold value. It is clearly illustrated that the maximum ratio is consistently obtained for block size values of 4–8.

The proposed algorithm is tested using more images with $X_{th}$ changing from 1 to 3 and a quantizer resolution of 3-bit. The simulation results show that less than 1 BPP can be achieved at a PSNR value of about 25 dB. Fig. 9(a) illustrates the PSNR to BPP ratio as a function of the block size $K$ as well as the threshold $X_{th}$. Fig. 9(b) illustrates the same results for a fixed threshold value of $X_{th} = 1$. From these simulation results, one can deduce that the optimum block size should be in the range of 4–8.

In order to illustrate the effectiveness of the proposed approach as compared to standard DPCM procedure, we performed extensive comparison using a large datasets. It is found that our proposed algorithm enables to improve the PSNR by at least 3 dB. Fig. 10 illustrates the comparison results of a standard DPCM procedure and our proposed approach for Lenna sample image. In the standard DPCM compression scheme, a 3-bit quantizer achieves a PSNR value of about 26 dB. In our proposed scheme, images are divided into 5 × 5 blocks and a 3-bit quantizer is used enabling us to achieve a PSNR value of 29.5 dB. This is explained by the fact that adding reference pixels in each block enables to reduce the error propagation as compared to a standard DPCM, and hence, improves the PSNR.

The proposed block-based compression scheme is followed by a quadrant tree decomposition scheme enabling further spatial redundancy removal, and hence, lower BPP is achieved. Furthermore, since pixel data are compressed prior to storage, a significant saving in memory is obtained.

IV. HARDWARE IMPLEMENTATION

A. Overview of the Overall System

Fig. 11(a) illustrates the architecture of the overall system. The sensor array is divided into 4 × 4 blocks. There are 16 pulselwidth modulation (PWM) [35], [36] DPSs used in each block. The detailed circuit of each pixel is shown in Fig. 11(c). The process starts with a reset phase in which the photodiode voltage $V_n$ is pulled up to $V_{dd}$. The voltage at node $V_n$ will be discharged proportionally to the light intensity. When $V_n$ reaches the threshold $V_{ref}$, the output of the comparator will trigger. In order to reduce the power consumption of the sensor array, a control signal is enabled once the comparator triggers. The power supply of the corresponding comparator will be cutoff until the image capture processing of the next frame begins. Within an elementary block, the brightest pixel will first trigger the comparator’s output and the first generated pulse is used as a start signal that controls the counting in the block-level counter. Once the start signal is generated, an 8-bit value is written into an 8-bit memory corresponding to the value of the brightest pixel. At the same time, a 2-bit block-level
counter will start counting once the start signal is received. The subsequent generated pulse will be used as a control signal that controls the write processing in the pixel-level memory in each pixel. Once a pixel fires, the 2-bit counter value is written into the corresponding pixel-level memory. The quantization step length \( \delta \) is controlled by the frequency of the clock used in the block-level counter. After capturing one frame of the image, the row/column select controllers will sequentially select one block to read out the value stored in the 8-bit memory (corresponding to the brightest pixel value) and all the differential values stored in the remaining 2-bit memories. In a PWM DPS array, \( V_{\text{fire}} \) of the brightest pixel will trigger first while that of the darkest pixel will trigger last. There is no need to retrieve the brightest pixel value in this proposed compression algorithm, which is, in fact, inherently retrieved in our PWM-based sensor. This greatly simplifies the implementation of the algorithm as no pixel values sorting is required. The first compression process is carried out during the image capture phase and the compressed results are recorded in the pixel-level memory. Overall, this scheme requires negligible overhead because simple processing is shared by the entire block (in this reported example, only 16 pixels need an overhead of 2-bit counter).

The simulated average current per pixel during a frame capture phase is approximately 1 \( \mu \)A. Considering a 64 \( \times \) 64 sensor array, the power consumption of the sensor array is approximated at around 10 mW. This power consumption is comparable to other DPS designs [31], [32]. Our power estimation reveals that the proposed sensor can achieve image acquisition/compression at power level of few hundred megawatts for mega pixel resolution.

### B. Memory Requirement Analysis

While comparing the proposed algorithm with a conventional architecture in which compression is performed after storing the full resolution, the processing overhead will be comparable since in both cases, compression will take place. To estimate the saving in terms of memory in our proposed scheme, let us assume in each \( K \times K \) block, the resolution of the quantizer is \( N \). In this case, there are \( K \times K \times N \)-bit pixel-level memory elements and one 8-bit block-level memory. The percentage of memory saved using our proposed algorithm is therefore expressed as

\[
\Delta \text{Mem\%} = \frac{(8 - N) \times K^2 - 8}{8 \times K^2} \times 100\%.
\]

Table III illustrates some examples of typical design scenarios and their corresponding memory saving, the proposed architecture greatly reduces the pixel-level memory requirement. Obviously, the larger the block size is, the greater the memory saving is. It should be noted that a 50% saving is possible even when using the smallest possible block size of \( (2 \times 2) \), while a block size of \( (8 \times 8) \) or larger enables more than 70% memory saving.

### V. IMPLEMENTATION AND TEST RESULTS

#### A. FPGA Implementation

In the previous section, it was shown that the processing involved in our compressive acquisition processing can be entirely implemented in digital. To validate the proposed processing, an FPGA platform is selected. The system is implemented into Xilinx Virtex-4 ML402 evaluation FPGA platform,[1] as illustrated in Fig. 12. A CMOS sensor is used to capture images, while a conversion circuit is used to mimic the output of the comparator \( V_{\text{fire}} \) within a 128 \( \times \) 128 time-domain image sensor array by generating digital pulsewidth-modulated

signals from the captured pixel value. The processing is then performed using digital logic circuits implemented on the FPGA platform. The compressed differential values stored in the N-bit pixel-level memories and the brightest pixel values with full precision stored in the 8-bit block-level memories are read-out and transferred to a PC through RS232 interface. The image decompression is performed offline using the PC. A clock frequency of 100 MHz is applied to the system, which is the maximum frequency available on the board.
B. Test Results

Testing is carried out on images captured using our experimental camera setup, as shown in Fig. 12, as well as some sample database images. Fig. 13 shows the reconstructed images for both sample database as well as real images acquired using our experimental platform. The resolution of the captured image is $128 \times 128$. A block size of $(4 \times 4)$ is used in our experimental platform, while 2-bit precision is selected for recording the differential pixel values. Data from the chip are transferred to the PC and the PSNR as well as the average number of bits per pixel are computed.

C. Layout

Fig. 14 illustrates the chip layout in standard 0.18 $\mu$m CMOS process. Table IV summarizes the performance of the proposed CMOS DPS array. The pixel layout of the proposed scheme is also compared to the pixel layout of a standard DPS, in which no pixel-level compression processing is involved, as illustrated in Fig. 15. A gain in pixel size of about 73% is achievable for a block size of 4. It is, however, important to note that in our proposed scheme, a block of $4 \times 4$ pixels requires an extra overhead of 2-bit counter. This lowers the pixel size gain to about 68% when compared to a standard DPS implementation. Table IV also compares the proposed design with some previously reported DPS architectures.

VI. CONCLUSION

In this paper, the concept of compressive acquisition CMOS image sensor is proposed and experimentally validated using FPGA implementation. The basic idea of compressing data online prior to storage is illustrated through a novel algorithmic solution, which is validated all the way through hardware implementation and experimental measurements. The paper illustrates the potential advantages of such a new design paradigm namely: 1) reduced silicon area required for the DPS; 2) improved fill-factor; and 3) compression processing integrated within the pixel array, enabling the concept of parallel processing. A mathematical model is derived enabling to optimize the quantization step length in order to improve the PSNR performance.
Extensive simulations were performed in order to evaluate the performance of the proposed concept and compare our proposed algorithm with a DPCM procedure. It was found that our proposed algorithm enables to improve the PSNR by at least 3-dB, while enabling lower BPP as the spatial redundancy is removed using block-based compression scheme. Results also illustrate that the proposed algorithm results in more than 50% on-chip memory saving even using the smallest possible block size of \((2 \times 2)\), while a block size of \((8 \times 8)\) or larger enables more than 70% memory saving. Since compression is performed prior to storage, these benefits would have direct impact on improving the performance of the DPS in terms of pixel size and fill-factor.

APPENDIX

PROOF OF THE RELATIONSHIP BETWEEN THE QUANTIZATION STEP LENGTH AND THE VARIANCE OF THE DISTRIBUTION

The choice of the quantization length \((\delta)\) as function of the number of quantization bits \((N)\) and the standard deviation of the Gaussian-like distribution \((\sigma)\) can be derived following the principle of minimizing the quantization error \(\varepsilon\) to \(\varepsilon_{\text{min}}\) as expressed in

\[
\varepsilon_{\text{min}} \Rightarrow \frac{\partial \varepsilon}{\partial \delta} = 0. \tag{A.1}
\]

Expressions (A.1) can be expressed as follows:

\[
\frac{\partial \varepsilon}{\partial \delta} = \frac{\delta}{\sum_{m=0}^{N-2} (n - m\delta)^2 P(n)} + \sum_{n=0}^{\delta} (n - (2^N - 1)\delta)^2 P(n)
\]

\[
= \frac{\delta}{\sum_{m=0}^{N-2} \int_{m\delta}^{(m+1)\delta} (x - m\delta - (2^N - 1)\delta)^2 f(x) dx} + \int_{(2^{N-1})\delta}^{\delta} (x - (2^N - 1)\delta)^2 f(x) dx
\]

\[
= (\delta - 1)^2 \sum_{m=0}^{N-2} f((m+1)\delta - 1)
\]

\[
- \sum_{m=0}^{N-2} f((m+1)\delta - 1) \cdot 2m(x - m\delta) f(x) dx
\]

\[
+ \int_{(2^{N-1})\delta}^{\delta} 2(2^N - 1)(x - (2^N - 1)\delta) f(x) dx
\]

\[
\approx (\delta - 1)^2 \sum_{m=0}^{N-2} f((m+1)\delta - 1) dm
\]

\[
- \sum_{m=0}^{N-2} f((m+1)\delta - 1) \cdot 2m(x - m\delta) f(x) dx
\]

\[
\approx (\delta - 1)^2 \int_{\delta}^{2^{N-1}\delta - 1} f(y) dy
\]

\[
- \int_{0}^{\delta} 2m(x f(x + m\delta) f(x) dx dm
\]

\[
\approx (\delta - 1)^2 \left( 1 - \int_{0}^{\delta} f(y) dy \right)
\]

\[
- \int_{0}^{\delta} \int_{x}^{x + 2m(x - x) f(x) dy dx dm
\]

\[
\approx \int_{0}^{\delta - 1} - \frac{2\delta}{\delta} \left[ \sigma^2 f(y) \right]_{x}^{\infty} - x \left( 1 - \frac{1}{2} e^{f(x)} \right) dx
\]

\[
+ \frac{(\delta - 1)^2}{\delta} \left( 1 - \frac{1}{2} e^{f(x)} \right)
\]

(A.2)

where the Gaussian error function \(erf(x)\) is twice larger than the integral of the normalized Gaussian distribution with 0 mean and variance of 0.5 as defined in

\[
erf(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt. \tag{A.3}
\]

Equation (A.3) can be approximated using Taylor series

\[
erf(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} = \frac{2}{\sqrt{\pi}} \left( x - \frac{x^3}{3} + \frac{x^5}{10} - \frac{x^7}{42} + \cdots \right). \tag{A.4}
\]

Expression (A.2) can be further expressed as

\[
\frac{\partial \varepsilon}{\partial \delta} \approx \frac{(\delta - 1)^2}{\delta} \left( 1 - \frac{1}{2} \left( \frac{\delta - 1}{\sqrt{2}\sigma} \right) \right)
\]

\[
- \int_{0}^{\delta - 1} \frac{2x}{\delta} \left[ - \sigma^2 f(x) - x + \frac{2}{\sqrt{2}} \left( \frac{x}{\sqrt{2}\sigma} \right) \right] dx
\]

\[
= \frac{(\delta - 1)^2}{\delta} - \frac{(\delta - 1)^2}{\sqrt{2}\sigma\delta} + \frac{\sigma^2}{\delta} \left( x + 2 \right)^\frac{\delta - 1}{3}\delta
\]

\[
\approx \frac{2(\delta - 1)^3}{\delta} - \frac{2\sigma^3}{\sqrt{2}\sigma\delta} - \frac{(\delta - 1)^4}{3\delta} - \frac{(\delta - 1)^4}{2\sqrt{2}\sigma\delta}
\]

\[
= \frac{1}{\sqrt{2}\sigma\delta} \left[ - (\delta - 1)^4 + \frac{4\sqrt{2}\sigma}{3} - 2 \right] (\delta - 1)^3
\]

\[
+ 2\sqrt{2}\sigma(\delta - 1)^2 - 4\sigma^4. \tag{A.5}
\]

Thus, expression (A.1) is rewritten as

\[
\frac{\partial \varepsilon}{\partial \delta} = 0 \Leftrightarrow \Phi(\delta - 1) = 0 \tag{A.6}
\]

with

\[
\Phi(x) = -x^4 + \frac{4\sqrt{2}\sigma}{3} - 2 x^3 + 2\sqrt{2}\sigma x^2 - 4\sigma^4.
\]

Equation (A.6) is a polynomial of degree 4. There is no direct solution to this expression, but iterative solution can be computed using MATLAB.

REFERENCES
