# A CMOS Image Sensor With On-Chip Image Compression Based on Predictive Boundary Adaptation and Memoryless QTD Algorithm

Shoushun Chen, *Member, IEEE*, Amine Bermak, *Senior Member, IEEE*, and Yan Wang, *Member, IEEE*

***Abstract*—This paper presents the architecture, algorithm, and VLSI hardware of image acquisition, storage, and compression on a single-chip CMOS image sensor. The image array is based on time domain digital pixel sensor technology equipped with nondestructive storage capability using 8-bit Static-RAM device embedded at the pixel level. The pixel-level memory is used to store the uncompressed illumination data during the integration mode as well as the compressed illumination data obtained after the compression stage. An adaptive quantization scheme based on fast boundary adaptation rule (FBAR) and differential pulse code modulation (DPCM) procedure followed by an online, least storage quadrant tree decomposition (QTD) processing is proposed enabling a robust and compact image compression processor. A prototype chip including 64 $\times$ 64 pixels, read-out and control circuitry as well as an on-chip compression processor was implemented in 0.35 $\mu$m CMOS technology with a silicon area of 3.2 $\times$ 3.0 mm$^2$ and an overall power of 17 mW. Simulation and measurements results show compression figures corresponding to 0.6–1 bit-per-pixel (BPP), while maintaining reasonable peak signal-to-noise ratio levels.**

***Index Terms*—CMOS image sensor, Hilbert Scan, on-chip image compression, quadrant tree decomposition (QTD).**

## I. Introduction

WITH THE development of network and multimedia technology, real time image acquisition and processing is becoming a challenging task because of higher resolution, which imposes very high bandwidth requirement. New applications in the area of wireless video sensor network and ultra low power biomedical applications have created new design challenges. For example, in a wireless video sensor network, limited by power budget, communication links among wireless sensor nodes are often based on low bandwidth protocols [1], such as ZigBee (up to 250 kb/s) and Bluetooth (up to 1 Mb/s). Even at the data rate of Bluetooth, conventional image sensor can barely stream an uncompressed 320 $\times$ 240 8-bit video at 2 frame/s. To avoid communication of raw data over wireless channels, energy efficient single chip solutions that integrate both image acquisition and image compression are required. Discrete wavelet transform (DWT), among various block-based transforms, is a popular technique used in JPEG-2000 image/video compression standard. However, implementation of image/video compression standards in cameras is computationally expensive, requiring a dedicated digital image processor in addition to the image sensor [2], [3]. A single chip solution is also possible by integrating compression functions on the sensor focal plane. This single-chip system integration offers the opportunity to reduce the cost, system size and power consumption by taking advantage of the rapid advances in CMOS technology. A number of CMOS image sensors with focal plane image compression have been proposed [4]–[11]. In [5], an 8 $\times$ 8 point analog 2-D-DCT processor is reported with fully switched capacitor circuits. In [6], floating gate technology is used to compute the DCT coefficients. However, the aforementioned designs do not actually implement compression on the focal plane since the entropy coding stage is located off-chip to limit chip size and cost. In [8], HAAR wavelets transforms are implemented by adopting a mixed-mode design approach to combine the benefits of both analog and digital domains. The CMOS image compression sensor features a 128 $\times$ 128 pixel array, a compression processor area of 1.8 mm$^2$ and a total chip area of 4.4 mm $\times$ 2.9 mm while the total power consumption was reported to be 26.2 mW [8]. In [9], a 44 $\times$ 80 CMOS image sensor integrating a complete focal-plane standard compression block with pixel prediction circuit and a Golomb–Rice entropy coder is reported. The chip has an average power consumption of 150 mW and a size of 2.596 mm $\times$ 5.958 mm in 0.35-$\mu$m CMOS technology. These various reported implementations are the results of tradeoffs between the level of complexity and functionalities of the focal-plane compression block and the associated silicon area and power consumption overheads for a given resolution of the imager. In [11], a compression processor is proposed, whose complexity and power consumption are to some extent independent of the resolution of the imager, making it very attractive for high resolution high-frame rate image sensors [11]. The single chip vision sensor integrates an adaptive quantization scheme followed by a quadrant tree decomposition (QTD) to further compress the image. The compression processor exhibits a significantly lower power consumption (e.g., 6.3 mW) and occupies a silicon area of 1.8 mm$^2$. The compression sensor

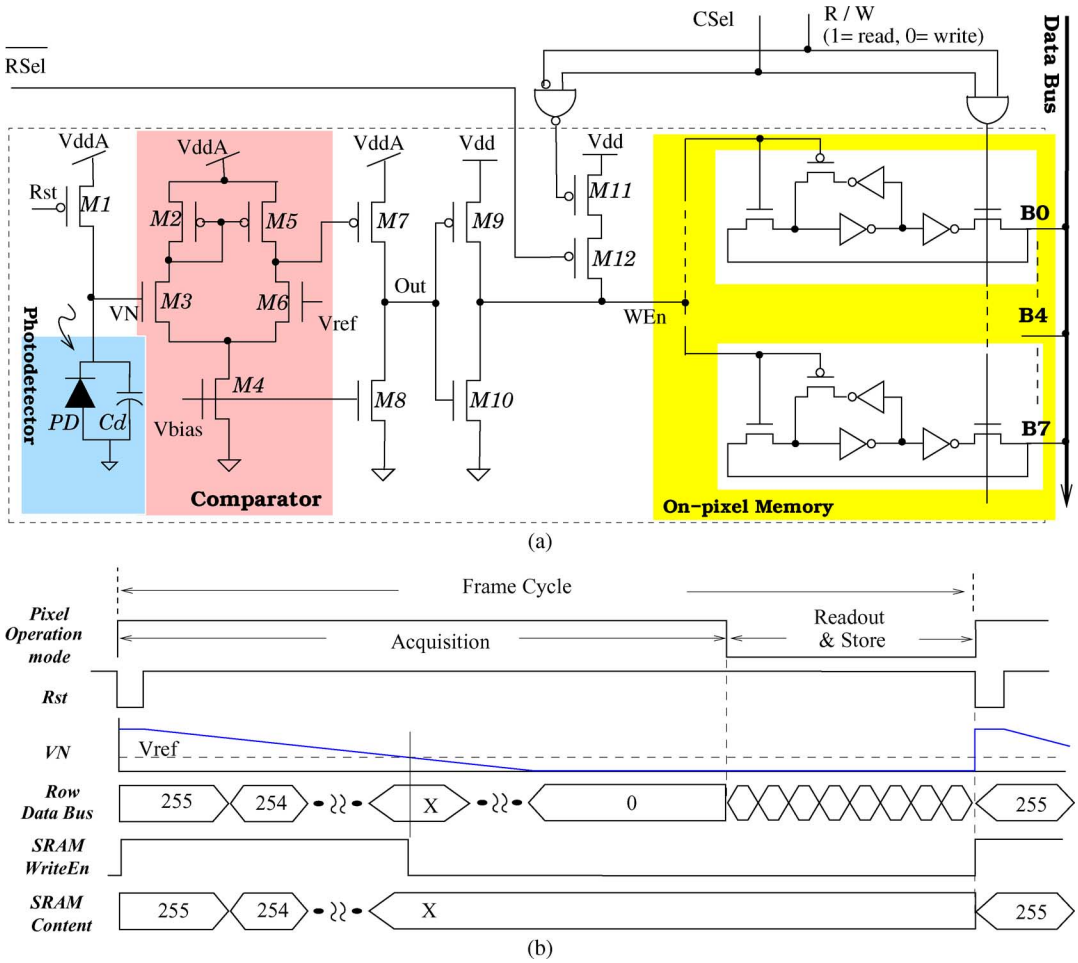



Fig. 1. (a) Pixel schematic illustrating the transistor-level circuitry of all the building blocks. (b) Pixel timing diagram showing the timing requirements for both acquisition and read-out modes.

permits to compress the data to 0.6–1 bit-per-pixel (BPP). The imager uses a Morton(Z) [12] block-based scanning strategy. The transition from one quadrant to the next involves jumping to a non-neighboring pixel, resulting in spatial discontinuities or image artifacts. In this paper, we propose a second generation prototype with the following main contributions: 1) new Hilbert scanning technique and its hardware implementation to avoid spatial discontinuities in the block-based scanning strategy; 2) the 1-bit fast boundary adaptation rule (FBAR) algorithm is performed on the predictive error rather than the pixel itself using differential pulse code modulation (DPCM), which results in improved performance; 3) introduction of memory reuse technique enabling over a threefold reduction in silicon area; and d) improved pixel structure for the DPS sensor. The proposed second generation imager with focal plane image compression is successfully implemented using Alcatel 0.35-$\mu$m CMOS technology.

The remainder of this paper is organized as follows. Section II introduces the design of a digital time-to-first-spike (TFS) image sensor. Section III discusses the algorithmic considerations for the FBAR algorithm combined with the predictive coding technique and presents the simulation results showing the improvements. Section IV describes the imager architecture and discusses design strategies used for implementing the Hilbert scan as well as the QTD processing involving the memory reuse concept. Section V reports the experimental results and provides a comparison with other compression processors. Section VI concludes this work.

## II. Pixel Design and Operation

The proposed system integrates the image sensor with pixel level ADC and frame storage together with the array-based standalone compression processor. The sensor array adopts a time domain digital pixel sensor (DPS) [13], in which the image is captured and locally stored at the pixel level. The image array consists of 64 $\times$ 64 digital pixel sensors. Fig. 1(a) illustrates the circuit diagram of the pixel, which includes four main building blocks, namely the photodetector *PD* with its internal capacitance $C_d$, followed by a reset transistor *M1*, a comparator (*M2–M8*), and an 8-bit SRAM. The comparator's output signal (*Out*) is buffered by (*M9–M10)* and then used as a write enable signal ("*WEn*") for the SRAM.

Fig. 1(b) illustrates the operation timing diagram of the proposed pixel, which is divided into two separate stages denoted as *Acquisition* stage and *Read-out/Store* stage. The first stage corresponds to the integration phase, in which the illumination level is recorded asynchronously within each pixel. The voltage of the sensing node *VN* is first reset to $VddA$. After that, the light

falling onto the photodiode discharges the capacitance $C_d$ associated with the sensing node, resulting in a decreasing voltage across the photodiode node. Once the voltage *VN* reaches a reference voltage $V_{\text{ref}}$, a pulse is generated at the output of the comparator *Out*. The time to generate the first spike is inversely proportional to the photocurrent [13] and can be used to encode the pixel's brightness. A global off-pixel controller operates as a timing unit, which is activated at the beginning of the integration process and provides timing information to all the pixels through "*Data Bus*". The pixel's "*WEn*" signal is always valid until the pixel fires. Therefore, the SRAM will keep tracking the changes on the "*Data Bus*" and the last data uploaded is the pixel's timing information. Once the integration stage is over, the pixel array turns to *Read-out/Store* stage. During this operating mode, the pixel array can be seen as a distributed static memory which can be accessed in both read or write modes using the Row and Column addresses. The on-chip image processor will first readout the memory content, compress the data and reuse the on-pixel memory as storage elements. With the external global control signal "*R/W*" and the row and column select signals $\overline{RSel}$ *and* $CSel$, the pixel's SRAM can be accessed in both read or write, namely:

- When the "*R/W*" signal is "1", the pixel will drive the "*Data Bus*" and the memory content will be readout.
- When the "*R/W*" signal turns to "0", transistor *M11* and *M12* will be turned on and the "WEn" signal is enabled again. The memory can therefore be accessed for write mode again and can be used as storage element for the processor.

This new feature differs significantly from previous DPS implementations, in which the on-pixel memory is only used for storing the raw pixel data. In our proposed design, the on-pixel memory is used to store the uncompressed illumination data during integration mode, as well as the compressed illumination data obtained after the compression stage. The memory is therefore embedded within the pixel array but also interacts with the compression processor for further processing storage. Moreover, the new pixel design also reduces the number of transistors from 102 to 84 compared to the pixel reported in [13]. This is achieved by removing the self-reset logic for the photodiode and the reset transistor for each bit of the on-pixel SRAM. In addition, the current pixel only requires two stages of inverter to drive the write operation for the memory. This is made possible because the SRAM's "*WEn*" signal is no longer pulse width sensitive.

## III. Image Compression—Algorithmic Considerations

The image compression procedure is carried-out in three different phases. In the first phase, the image data is scanned out off the array using Hilbert scanning then compared to a predictive value from a backward predictor. Based on the comparison result, a codeword (0 or 1) is generated and the comparison result is used as a feedback signal to adjust the predictor's parameters. In the second phase, the 1/0 codeword stream is considered as a binary image which is further compressed by the QTD processor. The compression information is encoded into a tree structure. Finally, the tree data together with non-compressed codewords are scanned out during the third phase.

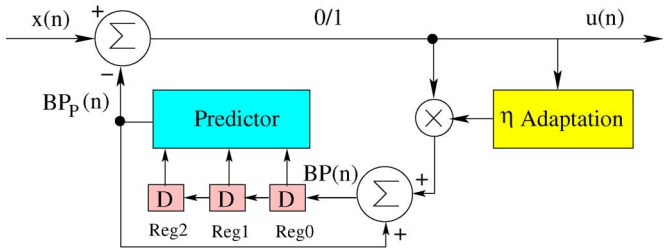



Fig. 2. 1-bit adaptive quantizer combined with DPCM.

### A. Predictive Boundary Adaptation

The proposed boundary adaptation scheme can be best described using an ordered set of boundary points $(BP)$ $y_0 < y_1 < \cdots < y_{i-1} < y_i < \cdots < y_{N-1} < y_N$ delimiting $N$ disjoint quantization intervals $R_1, \cdots, R_i, \cdots, R_N$, with $R_i = [y_{i-1}, y_i]$ [14]. The quantization process is a mapping from a scalar-valued signal $x$ into one of the reconstruction intervals, i.e., if $x \in R_j$, then $Q(x) = y_j$. Obviously, this Quantization process introduces quantization error when the number of quantization intervals is less than the number of bits needed to represent any element in a whole set of data. A $r$th power law distortion measure [14] can therefore be defined as

$$d(x, Q(x))\, D_r \equiv \sum_{i=1}^{N} |x - y_i|^r p(x) dx. \tag{1}$$

It has been shown that using FBAR [14] can minimize the $r$-th power law distortion, e.g., the mean absolute error when $r = 1$ or the mean square error when $r = 2$. At convergence, all the $N$ quantization intervals $R_i$ will have the same distortion $D_r(i) = D_r/N$ [14]. This property guarantees an optimal high resolution quantization. For a 1-bit quantizer, there will be just one adaptive boundary point $y$ delimiting two quantization intervals, with $R0 = [0, y]$ and $R1 = [y, 255]$. At each time step, the input pixel intensity will fall into either $R0$ or $R1$. $BP$ is shifted to the direction of the active interval by a quantity $\eta$. After that, the $BP$ itself is taken as the reconstructed value. With this adaptive quantization procedure, the $BP$ tracks the input signal and since $BP$ itself is used as the reconstructed value, a high resolution quantization is obtained even when using a single bit quantizer.

In our proposed system, when a new pixel $x(n)$ is read-out, its value is first estimated as $BP_P(n)$ through a backward predictor, as shown in Fig. 2. Three registers, denoted as $Reg0$, $Reg1$, $Reg2$ are used to store the history values of the previously reconstructed pixels. The $BP$ in our case is estimated as

$$BP_P = Reg0 \times 1.375 - Reg1 \times 0.75 + Reg2 \times 0.375. \tag{2}$$

Compared to the scheme reported in [11], $BP_P$ is now a function of three neighboring pixels and the estimated pixel value (prediction) is compared with the real incoming value. The comparison result, 0 or 1, is taken as a codeword $u(n)$, which is further used to update the boundary point

$$\text{if}\,(u(n) == 1)\,,\ BP = BP_P + \eta;\ \text{else}\ BP = BP_P - \eta. \tag{3}$$

The newly obtained $BP$ is feed back to update $Reg0$ and to predict the next pixel's value. The codeword $u(n)$ is also used to adjust another very important parameter $\eta$. Indeed, the adaptation step size parameter $\eta$ is found to affect the quantizer's performance [11]. On one hand, a large $\eta$ is preferred so as to track rapid fluctuations in consecutive pixel values. On the other hand, a small $\eta$ is preferred so as to avoid large amplitude oscillations at convergence. To circumvent this problem, we propose to make $\eta$ adaptive using a heuristic rule described as follows.

- *case1*: If the active quantization interval does not change between two consecutive pixel readings, we consider that the current quantization parameters are far from the optimum and $\eta$ is then multiplied by $\Lambda > 1$.
- *case2*: If the active quantization interval changes between two consecutive pixel readings, we consider that the current quantization parameters are near the optimum and thus $\eta$ is reset to its initial value $\eta_0$ (typically a small value).

This rule can be easily implemented by simply comparing two consecutive codewords, namely $u(n)$ and $u(n-1)$. Codeword values that are consecutively equal can be interpreted as a sharp transient in the signal as the $BP$ is consecutively adjusted in the same direction. In this situation, a large $\eta$ is used. Consequently, when $u(n) = u(n-1)$, $\eta$ is updated as $\eta = \eta \times \Lambda$ Otherwise, i.e., when $u(n) \neq u(n-1)$, $\eta = \eta_0$.

### B. Hilbert Scanning

The adaptive quantization process explained earlier permits to build a binary image on which QTD can be further employed to achieve higher compression ratio. The QTD compression algorithm is performed by building a multiple hierarchical layers of a tree which corresponds to a multiple hierarchical layers of quadrants in the array. To scan the image data out of the pixels array, many approaches can be employed. The most straightforward way is, for example, raster scan. However the choice of the scan sequence is very important as it highly affects the adaptive quantizer and QTD compression performance. Generally speaking, block based scan can result in higher peak signal-to-noise ratio (PSNR) and compression ratio because it provides larger spatial correlation, which is favorable for the adaptive quantization and QTD processing.

Fig. 3(a) illustrates a typical Morton (Z) scan [12] which is used to build the corresponding tree as reported in [11]. In this approach, transition from one quadrant to the next involves jumping to a non-neighboring pixel, which results in spatial discontinuity, which gets larger and larger when scanning the array due to the inherent hierarchical partition of the QTD algorithm. This problem can be addressed by taking the boundary point from the physically nearest neighbor of the previous quadrant rather than the previously scanned pixel [12]. Unfortunately, this solution comes at the expense of two additional 8-bit registers for each level of the quadrant. As shown in Fig. 3(a), two registers $(A4, B4)$ are needed to store the boundary point for the 4 $\times$ 4 quadrant level and two other registers $(A8, B8)$ are needed to store those related to the 8 $\times$ 8 quadrant level.

Fig. 3(b) illustrates an alternative solution using Hilbert scan sequence. In this scheme, multilayers hierarchical quadrants are sequentially read-out while maintaining spatial continuity during transitions from quadrant to the next. The storage

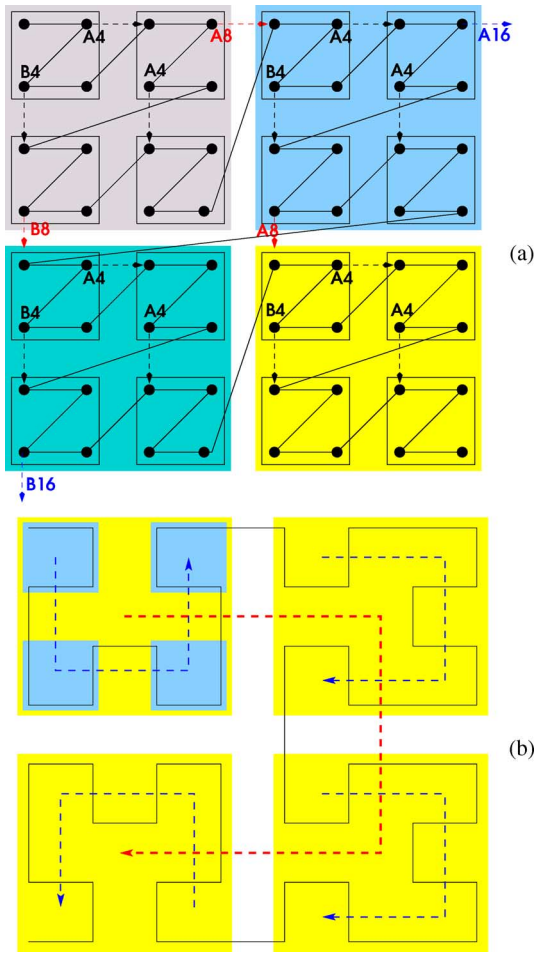



Fig. 3. (a) Boundary point propagation scheme using Morton (Z) scan [11]. When the Morton (Z) scan transits from one quadrant to another, instead of taking the boundary point from the previously scanned pixel, the boundary point is taken from the physically nearest neighbor of the previous quadrant. Implementing such scheme requires an extra two registers for each quadrant level. (b) Hilbert scan patterns at each hierarchy for a 8 $\times$ 8 array. One can note that, the scanning is also performed within multi-layers of quadrants (similar to Morton Z) but always keeping spatial continuity when jumping from one quadrant to another. This preserves the spacial neighborhood feature of the scan sequence and hence minimizes the storage requirement for the adaptive quantizer.

requirement issue is also addressed in this scheme as for the adaptive quantization processing, the neighboring pixel values are the ones just consecutively scanned. Hardware implementation of Hilbert scanning can be quite straightforward using hierarchical address mapping logic.

In summary, the compression scheme proposed in this paper can be generally interpreted as the cascade of two basic blocks namely the boundary adaptation block and the QTD processor. The first stage is a lossy compression for which there is a tradeoff between the compression ratio and the quality of the image. The compression performance is therefore controllable because the user can define the required number of bits at the output of the first block. The second stage (QTD processor) is a lossless compression as it processes a binary image and looks for removing spatial redundancy within the block. The compression quality in the second block is not controllable and is highly dependant on the input image. The main tradeoff involved in this design are related to the first stage in which

TABLE I

AVERAGE PERFORMANCE OF 20 TEST IMAGES FROM SCIENTIFIC IMAGE DATABASE [15] UNDER DIFFERENT OPERATING MODES, NAMELY DCT [16], WAVELET TRANSFORM [16], QTD [7], FIXED $\eta$ RASTER SCAN ($\eta_0$-R), ADAPTIVE $\eta$ RASTER SCAN ($\eta$-R), ADAPTIVE $\eta$ MORTON (Z) SCAN ($\eta$-MZ), ADAPTIVE $\eta$ SMOOTH BOUNDARY MORTON (Z) SCAN ($\eta$-SMOOTHMZ), ADAPTIVE $\eta$ HILBERT SCAN ($\eta$-HILBERT) AND ADAPTIVE $\eta$ WITH DPCM USING HILBERT SCAN ($\eta$-HILBERT+DPCM). $M = (PSNR/BPP)[dB/BPP]$. FOR EACH OPERATING MODE, $\eta_0$ WAS OPTIMIZED IN ORDER TO ACHIEVE THE BEST POSSIBLE PERFORMANCE. THE $\eta$-HILBERT+DPCM MODE PRESENTS THE BEST PSNR AND BPP FIGURES COMPARED TO THE FIRST GENERATION COMPRESSION ALGORITHM [11] AND FOR ALL POSSIBLE OPERATING MODES

| Operation modes | Size of test images | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $64 \times 64$ | | | | $128 \times 128$ | | | | $256 \times 256$ | | | | $512 \times 512$ | | | |
| | PSNR | BPP | M | $\eta_0$ | PSNR | BPP | M | $\eta_0$ | PSNR | BPP | M | $\eta_0$ | PSNR | BPP | M | $\eta_0$ |
| $\eta_0$-R | 21.00 | 1.01 | 20.87 | 26 | 22.14 | 0.95 | 23.43 | 22 | 23.66 | 0.88 | 26.87 | 18 | 25.34 | 0.84 | 30.15 | 15 |
| $\eta$-R | 21.15 | 1.07 | 19.79 | 21 | 22.42 | 0.97 | 23.05 | 20 | 24.01 | 0.93 | 25.95 | 16 | 25.93 | 0.91 | 28.52 | 12 |
| $\eta$-MZ | 21.48 | 0.97 | 22.15 | 18 | 22.82 | 0.89 | 25.64 | 16 | 24.32 | 0.84 | 28.78 | 13 | 25.90 | 0.85 | 30.55 | 11 |
| $\eta$-SmoothMZ | 22.37 | 0.96 | 23.37 | 17 | 23.83 | 0.90 | 26.51 | 14 | 25.49 | 0.86 | 29.79 | 12 | 27.45 | 0.89 | 30.76 | 9 |
| $\eta$-Hilbert | 22.77 | 0.98 | 23.11 | 18 | 24.26 | 0.93 | 26.05 | 14 | 26.05 | 0.88 | 29.70 | 12 | 28.11 | 0.99 | 28.27 | 7 |
| $\eta$-Hilbert+DPCM | **23.06** | 0.88 | 26.14 | 19 | **24.62** | 0.81 | 30.43 | 15 | **26.52** | 0.75 | 35.25 | 12 | **28.53** | 0.75 | 38.02 | 9 |
| DCT[16] | 19.28 | 0.91 | 21.18 | | 30.16 | 0.80 | 37.70 | | 33.98 | 0.70 | 48.54 | | 37.08 | 0.70 | 52.97 | |
| Wavelet[16] | 26.78 | 0.75 | 35.70 | | 33.01 | 0.75 | 44.01 | | 37.31 | 0.75 | 49.75 | | 39.84 | 0.75 | 53.12 | |
| QTD[7] | 31.49 | 6.56 | 4.80 | | 30.34 | 6.13 | 4.95 | | 30.10 | 5.70 | 5.28 | | 28.88 | 5.23 | 5.52 | |

the number of bits at the output of the adaptive quantizer. A larger number of bits enables improved signal to noise ratio and better image quality but obviously at the expense of increased complexity, increased BPP as well as increased power consumption. In terms of scalability of the architecture, it should be noted that the boundary adaptation block is completely independent upon the array size and is performed on the fly while scanning out the raw data, therefore, it is highly scalable. The QTD computations however involve a top down (tree construction) and a bottom up (tree trimming) processing. The QTD processing is therefore not scalable. Increasing the size of the imager would require redesigning the QTD processor, but since the QTD algorithm is quite structural, it is not difficult to scale the HDL code.

### C. Simulation Results

The performance of our proposed compression scheme, i.e., adaptive $\eta$ with DPCM using Hilbert scan ($\eta$-Hilbert+DPCM), is compared with other operating modes, namely fixed $\eta$ raster scan ($\eta_0$-R), adaptive $\eta$ raster scan ($\eta$-R), adaptive $\eta$ Morton (Z) scan ($\eta$-MZ), adaptive $\eta$ smooth boundary Morton (Z) scan ($\eta$-SmoothMZ) [11], adaptive $\eta$ Hilbert scan ($\eta$-Hilbert) for a set of test images from scientific image database [15], as illustrated in Table I. For each sample image, different resolutions are generated ($64 \times 64$, $128 \times 128$, $256 \times 256$, $512 \times 512$,) and used in our comparative study. Simulation on different resolutions is important because our hardware implementation is low resolution and the aim of this simulation is to provide some insights on how the processor will perform if we were to increase the resolution. Obviously, the performance of the quantizer is highly dependent on the choice of $\eta$ [11]. There even exists an optimal $\eta$ value for a particular image under each operation mode. However, it is unpractical to tune fine $\eta$ in order to obtain "optimal" performance for each input image. Therefore, for each operation mode reported in Table I, we sweep the value of $\eta$ from 5 to 35 and calculate the average PSNR and BPP to find the optimal performance for the whole data set. Fig. 4 reports the results when sweeping $\eta$ (5–35) for the "DPCM using Hilbert scan" ($\eta$-Hilbert+DPCM) configuration. Both PSNR and BPP are highly dependant upon the value of $\eta$. Using a large value for $\eta$ enables faster tracking of sharp transients in the signal and hence improved compression ratios are obtained when combined with QTD. However $\eta$ cannot be increased indefinitely as it will result in a rapidly degraded PSNR performance. For image sizes of $256 \times 256$ and $512 \times 512$, the optimum $\eta$ was found to be around 13 and 10, respectively. From Table I, one can notice the benefit of adaptive $\eta$ by comparing the performance figures in the first and second rows, where both modes are based on conventional row and column raster scan. With adaptive $\eta$ (second row), the PSNR is increased by about 0.4 dB. Morton (Z) enables better performance as compared to raster scan because it is a block based scan improving the spatial correlation, which is exploited by both the adaptive Q and QTD processing blocks (third row). However, in Morton (Z) scan, the transition from one quadrant to the next involves transitions to a non-neighboring pixel resulting in spatial discontinuity. In [11], a smooth boundary point propagation scheme is proposed, enabling to solve this spatial discontinuity issue resulting in a PSNR improvement of about 1–1.5 dB (fourth row). Hilbert scan provides another interesting block-based scan strategy featuring spatial continuity. It is clearly shown that the performance of Hilbert scan are superior to that of raster, Morton Z and even smooth Morton Z scanning strategies (fifth row). It is important to note from this table that using predictive boundary adaptive coding combined with Hilbert scanning ($\eta$-Hilbert+DPCM) enables about 25% improvement in terms of performance (expressed by the PSNR to BPP ratio) compared to the first generation design [11]. From Table I, we can also note that performance improvements are obtained when using large size images. For our proposed algorithm ($\eta$-Hilbert+DPCM), using $512 \times 512$ format instead of $64 \times 64$ enables a 23% and a 14% improvements in terms of PSNR and BPP, respectively. This represents a significant improvement suggesting that the proposed algorithm is much more effective for large size images. Table I also illustrates a comparison of the proposed algorithm to other standards. One can note that the performance of our processor are clearly superior to a standalone QTD and comparable to DCT-based compression DCT [16] but clearly inferior to that of wavelet based compression DCT [16]. It is however important to note that the hardware complexity is an order of magnitude simpler when compared to both DCT and wavelet-based compression. This is due to the inherent advantage of boundary adapta-

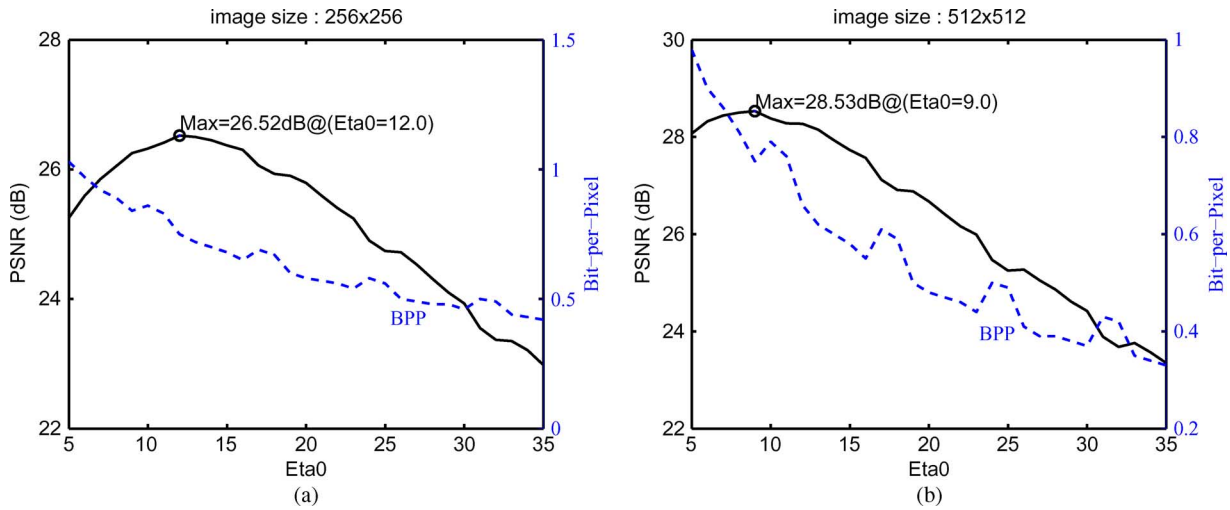



Fig. 4. Simulation results illustrating the compression performance (PSNR and BPP) as function of $\eta 0$. The left and right $y$-axes illustrate the PSNR and BPP, respectively. The simulation is reported for two image sizes namely: (a) image size of 256 $\times$ 256 and (b) image size $= 512 \times 512$.

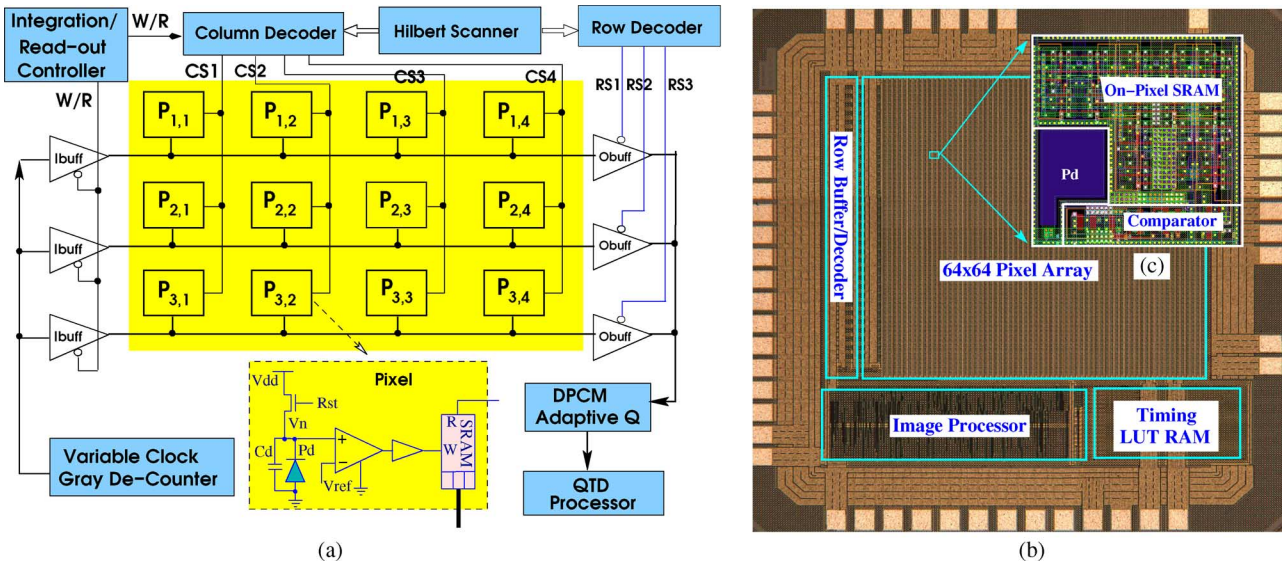



Fig. 5. (a) Architecture of the overall imager including the sensor and the processor. (b) Corresponding microphotograph of the chip implemented in Alcatel 0.35-$\mu$m CMOS technology with the main building blocks highlighted. (c) Layout of the pixel.

tion processing requiring simple addition, subtraction, and comparison for $\eta$ adaptation. The storage requirement is however quite demanding for QTD processing since a tree construction and storage is required, however, this issue and some of the hardware optimization techniques will be addressed in our proposed system, as will be explained in Section IV.

## IV. VLSI Implementation

### A. Imager Architecture

Fig. 5(a) shows the block diagram of the overall system featuring the CMOS image sensor integrated together with the compression processor including the adaptive DPCM quantizer and the QTD processor. The image array consists of 64 $\times$ 64 digital pixel sensors. The pixel array is operated in two separate phases. The first phase corresponds to the integration phase, in which the illumination level is recorded and each pixel sets its own integration time which is inversely proportional to the photocurrent. A timing circuit is used in order to compensate for this nonlinearity by adjusting the quantization times using a nonlinear clock signal which is fed to the counter [13]. Moreover, proper adjustment of the quantization timing stamps stored in a 16 $\times$ 256-bit on-chip SRAM memory enables to implement various transfer functions including a log-response [17].

During the integration phase, the row buffers drive the timing information to the array, using gray code format. Once the longest permitted integration time is over, the imager turns into the read-out mode. The row buffers are disabled and the image processor starts to operate. First, the QTD processor will generate linear quadrant address which is then translated into Hilbert scan address by the Hilbert Scanner block. The address is decoded into "Row Select Signal $(RSel)$" and "Column Select Signal $(CSel)$". The selected pixel will drive

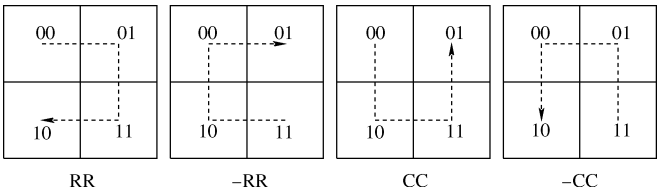



Fig. 6. Basic scanning patterns found in Hilbert scan.

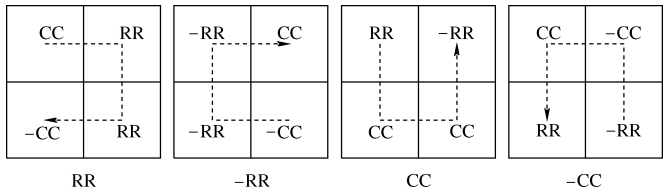



Fig. 7. Hilbert scanning patterns between hierarchies of quadrants. If the parent scanning pattern is $RR$, its four children quadrants' pattern are $CC \mapsto RR \mapsto RR \mapsto -CC$, respectively.

the data bus and its value will be first quantized by the DPCM Adaptive Quantizer then the binary quantization result will be compressed by the QTD processor.

### B. Hilbert Scanner

Hilbert scanning is actually composed of multiple levels of four basic scanning patterns as shown in Fig. 6.

These are denoted as $RR$, $-RR$, $-CC$, and $CC$, respectively. $RR$ represents a basic scanning pattern featuring a relationship between its linear scanning sequence and the physical scanning address described as follows:

$$RR \begin{cases} \text{Linear Add: } ('b00) \rightarrow ('b01) \rightarrow ('b10) \rightarrow ('b11) \\ \Downarrow \\ \text{Hilbert Add: } ('b00) \rightarrow ('b01) \rightarrow ('b11) \rightarrow ('b10). \end{cases}$$

$CC$ represents another basic scanning pattern with the following address mapping relationship:

$$CC \begin{cases} \text{Linear Add: } ('b00) \rightarrow ('b01) \rightarrow ('b10) \rightarrow ('b11) \\ \Downarrow \\ \text{Hilbert Add: } ('b00) \rightarrow ('b10) \rightarrow ('b11) \rightarrow ('b01). \end{cases}$$

For an array of $2^m \times 2^m$ pixels, the whole Hilbert scan can be represented by $m$ levels of scanning patterns. For an intermediate level, its scanning pattern is determined by its parent quadrant's pattern. At the same time, its scanning pattern can also determine its child quadrants' patterns, as illustrated in Fig. 7. If a quadrant is in the $RR$ format, then its four children quadrants must be in the $CC \mapsto RR \mapsto RR \mapsto -CC$ formats, respectively. Using this strategy, it is possible to implement Hilbert scanning in a top-down approach. Firstly, a linear address is used to segment the whole array into quadrant levels. Each quadrant level is addressed by a 2-bit address. Second, the scanning pattern for each quadrant level is retrieved. For the very top quadrant level, the scanning sequence is predefined as either $RR$ or $CC$. If the current scan sequence is $RR$, then the scanning sequences of the four children quadrants should be $CC \mapsto RR \mapsto RR \mapsto -CC$, respectively. The two most significant bits (MSBs) of the address are used to decode one out of four largest quadrants being scanned. If the 2-bit MSB are equal to 'b11, the fourth quadrant is being scanned and its scanning pattern is set to $-CC$ format. Consequently, its four sub-quadrants are set to be $-RR \mapsto -CC \mapsto -CC \mapsto RR$ formats, respectively. Furthermore the decoding of the sub-quadrants is performed using the second 2 MSB bits of the linear address. Applying the same procedure on the subsequent hierarchical levels enables the mapping of all the linear address into Hilbert scan address. The above mapping only involves bitwise manipulation and therefore, no sequential logic is needed, which results in very compact VLSI implementation.

### C. QTD Algorithm With Pixel Storage Reuse

For our 64 × 64 array, the tree information is to be stored in registers with a total number of $1024+256+64+16+4+1 = 1407$. In [11] the QTD tree is built out of the pixel array, which occupies significant silicon area. A possible solution to save area is based on the following observation: The proposed 1-bit FBAR algorithm compresses the original 8-bit pixel array into a binary image with only 1-bit per pixel. QTD tree can therefore be stored inside the array by reusing the storage elements of the DPS pixels.

The QTD algorithm is based on the fact that if a given quadrant can be compressed, only its first pixel's value and its root are necessary information. All the other pixels in the quadrant and the intermediate level nodes on the tree can be compressed. The only storage requirement outside the pixel array is a 12-bit shift register used to temporarily store the nodes of the current quadrant level. For the sake of clarity, let us look at the operating principle of one intermediate level as shown in Fig. 8. Each valid bit of the shift register SR4 represents the compression information of a 4 × 4 block. During the scanning phase, each time a 4 × 4 block is scanned, the shift register SR4 will shift in a new value $(new\_node\_4 \times 4)$. However, each time the higher level block (8 × 8 block) is scanned and if this 8 × 8 block can be compressed, the last 4 bits of SR4 will be shifted out. This principle can be described as: "*a lower level block is dropped if its parent can be compressed*". When the SR4 register is full (12 bits), the previous 8 bits correspond to the nodes that can't be compressed and will be written back to a special location of the array, which is at the lower right corner of the corresponding quadrant. For example, the SR4 register can only be stored at the binary addresses of $'bxx\_ss\_11\_11$, where $ss$ can be $'b00$, $'b01$ or $'b10$ and $xx$ can be $'b000000$ to $'b111111$. While at the lowest pixel level, a 26-bit shift register $(SRPix)$ is maintained to store the first pixel of each quadrant. If the 2 × 2 level quadrant can be compressed, the last 3 bits of $SRPix$ will be shifted off and if the 4 × 4 level quadrant can be compressed, the last 6 bits of $SPPix$ will be shifted out, etc.,. . .. If it is full, the previous 8 bits will be written back into the array at the address location of $b'xx\_ss$, where $ss$ is $'b00$, $'b01$ or $'b10$.

### D. Physical Implementation

The single chip image sensor and compression processor is implemented using 0.35 $\mu$m Alcatel CMOS digital process (1-poly 5 metal layers). Fig. 5(a) illustrates the architecture of the overall imager including the sensor and the processor. Fig. 5(b) illustrates the corresponding microphotograph of the chip with a total silicon area of 3.2 × 3.0 mm$^2$. The 64 × 64 pixel array was implemented using a full-custom approach. The main building blocks of the chip are highlighted in Fig. 5(b).

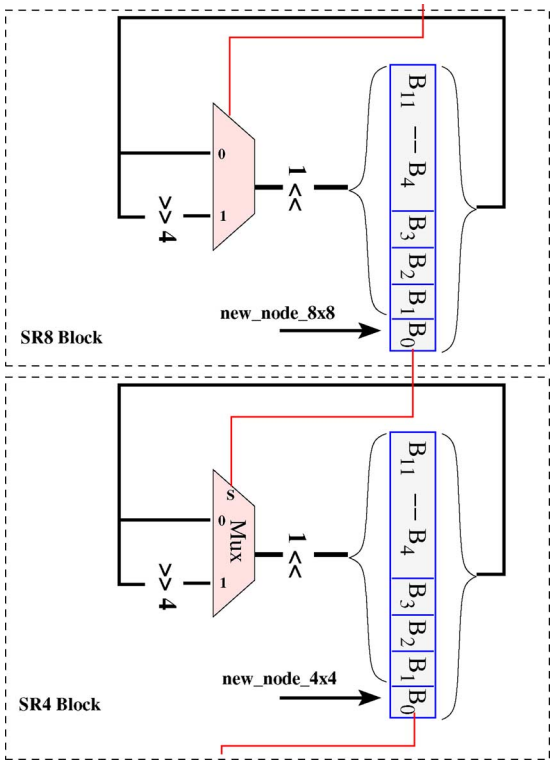



Fig. 8. Block diagram of the shift register at the $4 \times 4 (SR4)$ and $8 \times 8 (SR8)$ block level. At each level, the 4-bit LSB will be shifted off if its higher level's lowest bit is "1", which means its higher level can be compressed. In other words, one will be kicked out if its parent can be compressed.

TABLE II
NUMBER OF FLIP-FLOPS USED IN THIS WORK AND [11]. NA STANDS FOR NOT APPLICABLE

| Functional Block | This work | [11] |
|---|---|---|
| Adaptive $\eta$ | 9 | 9 |
| DPCM | 24 | NA |
| Smooth MZ | NA | 64 |
| QTD | 202 | 1407 |
| Hilbert Scan | 0 | NA |
| **Total** | **235** | **1480** |

The photosensitive elements are $n^+p$ photodiodes chosen for their high quantum efficiency. Except for the photodiode, the entire in-pixel circuitry [see Fig. 1(a)] is shielded from incoming photons to minimize the impact of light-induced current resulting in parasitic light contribution to the signal. Guard rings are extensively used to limit substrate coupling and as means to shield the pixels from the outer array digital circuitry. Power and ground buses are routed using top layer metal. Fig. 5(c) illustrates the layout of the pixel. Each pixel occupies an area of 39 $\times$39 $\mu$m$^2$ with a fill-factor of 12%. The digital processor was synthesized from HDL and implemented using automatic placement and routing tools. The digital processor occupies an area of $0.25 \times 2.2 = 0.55$ mm$^2$. It should be noted that the newly proposed design achieves an area reduction of over 70% as compared to [11] (1.8 mm$^2$). This is mainly due to the optimization of the storage requirement for the QTD tree using "Pixel Storage Reuse" technique, which saves a large number of flip-flops. Table II, reports the number of flip-flops used in this processor compared to that reported in [11].

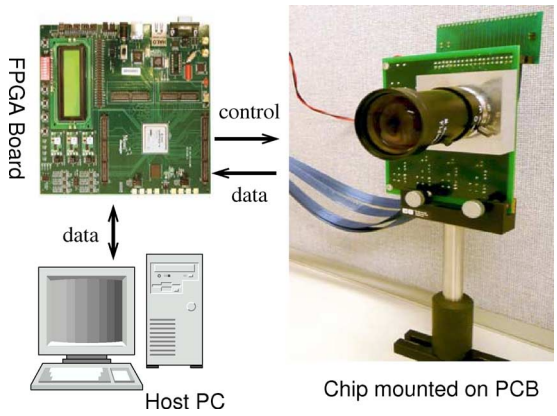



Fig. 9. FPGA-based test platform which is composed of a host computer, FPGA board (Memec Virtex-4 MB) and the chip under test.

TABLE III
SUMMARY OF THE CHIP PERFORMANCE

| | |
|---|---|
| Technology | Alcatel 0.35$\mu$m CMOS<br>5 metal single-poly, twin well |
| Architecture | Array-level adaptive/QTD Compression |
| Quantization bits | 8-bits |
| Array size | 64×64 |
| Chip area | 3.2×3.0mm$^2$ |
| Image processor area | 2.2×0.25mm$^2$ |
| Pixel area | 39×39$\mu$m$^2$ |
| Fill factor | 12% |
| FPN | 0.8% |
| Dynamic range | >100dB |
| Power supply | 3.3V |
| Power consumption (chip) | 17mW |
| Power consumption (proc.) | 2mW |

## V. EXPERIMENTAL RESULTS AND COMPARISON

In order to characterize the prototype, a field-programmable gate-array (FPGA)-based testing platform has been developed shown in Fig. 9. The test chip was mounted on a printed circuit board outfitted with an FPGA platform and a universal asynchronous receiver/transmitter (UART) connection for communications with a PC that acts as the decoding platform. The compressed bit stream is sent to the PC and is decoded on software using the inverse predictive adaptive quantization and the QTD coding algorithms. The FPGA is configured to provide the input control signals and temporarily store the output signals from the prototype. The SRAM of the timing unit is first configured followed by a global pixel reset signal, which starts the integration process. The timing unit decounts from "255" to"0", in gray code format. When it reaches the value of "0", i.e., the darkest gray level value, the integration process is completed and the image processor is enabled. The FPGA temporarily stores the captured data into an on-board SRAM and then sends it to a host computer through a UART connection. As described earlier, the imager will send the trimmed tree data followed by the compressed binary image data (quantization codewords), which is actually the first pixel within each compressed quadrant. As a result, on the host computer, the same tree is first rebuilt and the whole array can be reconstructed based on the received tree topology and the first pixel value of each quadrant. Table III summarizes the performance of the chip.

TABLE IV
COMPARISON OF OUR DESIGN WITH SOME IMAGERS WITH ON-CHIP COMPRESSION REPORTED IN THE LITERATURE. THESE DESIGNS ARE BASED ON DIFFERENT COMPRESSION SCHEME SUCH AS DCT, WAVELET TRANSFORM, PREDICTIVE CODING. ESTIMATED AREAS ARE MARKED IN ASTERISK $(*)$

| Compression scheme | DCT [6] | QTD [7] | Haar Wavelet [8] | Predictive [9] | SPIHT [10] | AQ /QTD [11] | This work |
|---|---|---|---|---|---|---|---|
| Architecture | pixel and chip level | Column level | Column level | Column level | Chip level | Chip level | Chip level |
| Compression Type | Lossy | Lossy | Lossy | Lossless | Lossy | Lossy | Lossy |
| Technology | 0.5$\mu$m | 0.35$\mu$m | 0.35$\mu$m | 0.35$\mu$m | 0.5$\mu$m | 0.35$\mu$m | 0.35$\mu$m |
| Array Size | 104×128 | 32×32 | 128×128 | 80×44 | 33×25 | 64×64 | 64×64 |
| Processor Area | 1.5mm$^2$* | 0.4mm$^2$ * | 1.8mm$^2$ | 0.11mm$^2$ | 0.36mm$^2$* | 1.8mm$^2$ | 0.55mm$^2$ |
| Power | 80$\mu$W/frame | 70mW/chip | 26.2mW/chip 24.4mW/proc. | 150mW/chip 3mW/proc. | 0.25mW/chip | 20mW/chip 6.3mW/proc. | 17mW/chip 2mW/proc. |
| post-proc requirement | Yes | No | No | No | Yes | No | No |

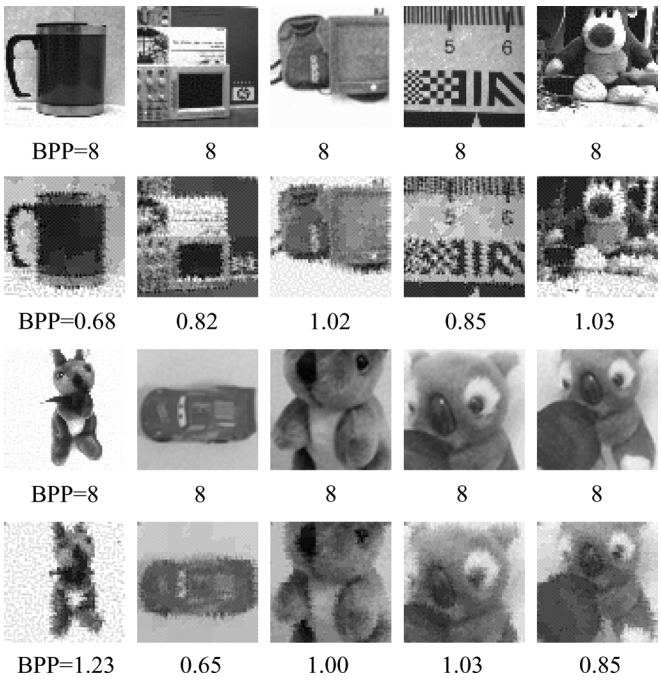



Fig. 10. Captured images from the prototype chip. The first and third rows show the sample images captured without compression while the second and fourth rows represent the reconstructed compressed images using the proposed image compression processor.

The chip was tested in both compressing and noncompressing modes and consumes about 17 mW power, in which about 15 mW is consumed by the sensor array and 2 mW is consumed by the image processor. Fig. 10 shows some sample $64 \times 64$ 8-bit sample images as well as compressed sample images with their corresponding BPP figure. For the compressing modes, the data from the CMOS image sensor are acquired using the FPGA platform and transferred to the PC for display. Once the data is received, the total number of bits per frame $(B_F)$ is counted and the BPP is calculated as

$$BPP = \frac{B_F}{64 \times 64}. \tag{4}$$

Table IV compares the performance of the our proposed scheme presented in this paper with the first generation processor [11] as well as other imagers with compression processors reported in the literature [6]–[10]. One should note that the comparison of different compression processors is not obvious as the target performance is different for different designs and therefore computational requirements and circuit complexities, image quality and compression performance as well as imager resolution and specifications may vary significantly. In addition, some designs implement only certain building blocks of the compression algorithm on the focal plane, while an external post-processing is still required to realize a full compression system. Some other implementations only focus on the compression processing ignoring the sensor, the ADC circuitry and the frame storage and buffering. This renders the comparison of different designs very subjective and nonconclusive. One can however notice that our proposed chip does not require any post-processing and the compression processor is successfully integrated together with the sensor achieving quite low silicon area and reasonably low power consumption.

## VI. CONCLUSION

This paper reports a single chip CMOS image sensor with on-chip image compression processor, based on a hybrid predictive boundary adaptation processing and QTD encoder. Hilbert scan is employed to provide both spatial continuity and quadrant based scan. The proposed compression algorithm enables about 25% improvement in terms of performance (PSNR to BPP ratio) compared to the first generation design. Reported performance are clearly superior to that of a standalone QTD and quite comparable to DCT-based compression. The hardware complexity is however an order of magnitude simpler when compared to both DCT and wavelet based compression. This is due to the inherent advantage of boundary adaptation processing requiring simple addition, subtraction, and comparison for $\eta$ adaptation. The storage requirement is however quite demanding for QTD processing since a tree construction and storage is required, however, this issue is addressed in this paper by introducing a QTD algorithm with pixel storage reuse technique. The memory is therefore embedded within the pixel array but also interacts with the compression processor for further processing storage. This technique has enabled an area reduction of the compression processor by about 70%. The proposed hardware friendly algorithm has therefore enabled a complete system implementation which integrates the image sensor with pixel level ADC and frame storage together with the full standalone compression processor including predictive boundary adaptation and QTD. A prototype chip including a $64 \times 64$ pixel array was successfully implemented in 0.35-$\mu$m CMOS technology with a silicon area of $3.2 \times 3.0$ mm$^2$. A very interesting fact about this design is that compression is performed on-the-fly while scanning out the data using Hilbert scanner. This results in reduced timing overhead while the overall system consumes less than 18 mW of power.

## ACKNOWLEDGMENT

The authors would like to thank Dr. D. Martinez for helpful discussions.

## REFERENCES

[1] G. Pekhteryev, Z. Sahinoglu, P. Orlik, and G. Bhatti, "Image transmission over ieee 802.15.4 and zigbee networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, vol. 4, pp. 3539–3542.

[2] K.-Y. Min and J.-W. Chong, "A real-time JPEG encoder for 1.3 Mega pixel CMOS image sensor SOC," in *Proc. IEEE 30th Annu. Conf. Ind. Electron. Soc.*, 2004, vol. 3, pp. 2633–2636.

[3] L.-G. Chen, J.-Y. Jiu, H.-C. Chang, Y.-P. Lee, and C.-W. Ku, "A low power 2-D DCT chip design using direct 2-D algorithm," in *Proc. Asia South Pacific Des. Autom. Conf.*, 1998, pp. 145–50.

[4] Q. Luo, J. Harris, Z. Sahinoglu, P. Orlik, and G. Bhatti, "A novel integration of on-sensor wavelet compression for a CMOS imager," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2002, vol. 3, pp. 325–328.

[5] S. Kawahito, M. Yoshida, M. Sasaki, K. Umehara, D. Miyazaki, Y. Tadokoro, K. Murata, S. Doushou, and A. Matsuzawa, "A CMOS image sensor with analog two-dimensional DCT-based compression circuits for on-chip cameras," *IEEE J. Solid-State Circuits*, vol. 32, no. 12, pp. 2030–2041, Dec. 1997.

[6] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "Matia: A programmable 80 $\mu$W/frame CMOS block matrix transform imager architecture," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, Mar. 2006.

[7] E. Artyomov and O. Yadid-Pecht, "Adaptive multiple-resolution CMOS active pixel sensor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 10, pp. 2178–2186, Oct. 2006.

[8] A. Nilchi, J. Aziz, and R. Genov, "Focal-plane algorithmically-multiplying CMOS computational image sensor," *IEEE J. Solid-State Circuits*, vol. 44, no. 6, pp. 1829–1839, Jun. 2009.

[9] W. D. Len-Salas, S. Balkir, K. Sayood, N. Schemm, and M. W. Hoffman, "A CMOS imager with focal plane compression using predictive coding," *IEEE J. Solid-State Circuits*, vol. 42, no. 11, pp. 2555–2572, Nov. 2007.

[10] Z. Lin, M. W. Hoffman, N. Schemm, W. D. Leon-Salas, and S. Balkir, "A CMOS image sensor for multi-level focal plane image decomposition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 9, pp. 2561–2572, Oct. 2008.

[11] S. Chen, A. Bermak, W. Yan, and D. Martinez, "Adaptive-quantization digital image sensor for low-power image compression," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 1, pp. 13–25, Jan. 2007.

[12] E. Artyomov, Y. Rivenson, G. Levi, and O. Yadid-Pecht, "Morton (Z) scan based real-time variable resolution CMOS image sensor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 15, no. 7, pp. 947–952, Jul. 2005.

[13] A. Kitchen, A. Bermak, and A. Bouzerdoum, "A digital pixel sensor array with programmable dynamic range," *IEEE Trans. Electron Devices*, vol. 52, no. 12, pp. 2591–2601, Dec. 2005.

[14] D. Martinez and M. M. V. Hulle, "Generalized boundary adaptation rule for minimizing $r^{th}$ power law distortion in high resolution quantization," *Neural Netw.*, vol. 8, no. 6, pp. 891–900, 1995.

[15] Signal and Image Processing Institute, University of Southern California, Los Angeles, CA, "The USC-SIPI image database," 2009. [Online]. Available: http://sipi.usc.edu/database/index.html

[16] Delft University of Technology, Delft, The Netherlands, "VcDemo software," 2009. [Online]. Available: http://ict.ewi.tudelft.nl

[17] A. Bermak and A. Kitchen, "A novel adaptive logarithmic digital pixel sensor," *IEEE Photon. Technol. Lett.*, vol. 18, no. 20, pp. 2147–2149, Oct. 2006.

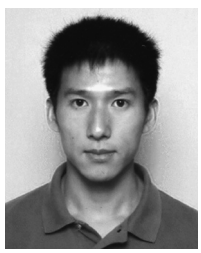
**Shoushun Chen** (M'08) received the B.S. degree from Peking University, Peking, China, the M.E. degree from Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong, in 2000, 2003 and 2007, respectively.

He held a Postdoctoral Research Fellowship with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology for one year after graduation. From February 2008 to May 2009, he was a Postdoctoral Research Associate with the Department of Electrical Engineering, Yale University. In July 2009, he joined Nanyang Technological University, Singapore, as an Assistant Professor. He has been actively involved in a number of research projects related to VLSI implementation of microprocessor, circuits and systems design for vision sensor, algorithmic design for image processing. In Chinese Academy of Sciences, he was a Key Backend Designer of "Loongson-1" CPU which is the first general purpose CPU designed in China. In Hong Kong, his research was mainly related to design and VLSI implementation of ultra-low power and wide dynamic range CMOS image sensors using time encoding techniques and asynchronous readout strategies. At Yale, he proposed a bio-inspired human posture recognition algorithm and his work was toward a combination of smart vision sensors and energy-efficient algorithm. His research interests include mixed mode integrated circuits design for sensors, feature extracting biomimetic sensors for sensor networks, energy-efficient algorithms for object recognition, smart vision sensors, and asynchronous VLSI circuits and systems.

**Amine Bermak** (M'99–SM'04) received the M.Eng. and Ph.D. degrees in electronic engineering from Paul Sabatier University, Toulouse, France, in 1994 and 1998, respectively.

During his Ph.D., he was part of the Microsystems and Microstructures Research Group at the French National Research Center LAAS-CNRS where he developed a 3-D VLSI chip for artificial neural network classification and detection applications. He then joined the Advanced Computer Architecture Research Group, York University, York, U.K., where he was working as a Postdoctoral Researcher on VLSI implementation of CMM neural network for vision applications in a project funded by British Aerospace. In 1998, he joined Edith Cowan University, Perth, Australia, first as a Research Fellow working on smart vision sensors, then as a Lecturer and a Senior Lecturer in the School of Engineering and Mathematics. He is currently an Associate Professor with the Electronic and Computer Engineering Department, Hong Kong University of Science and Technology (HKUST), Hong Kong, where he is also serving as the director of Computer Engineering and the Director of M.Sc. degree in Integrated Circuit Design (ICDE). His research interests include VLSI circuits and systems for signal, image processing, sensors and microsystems applications. He has published extensively on the above topics in various journals, book chapters and refereed international conferences.

Dr. Bermak was a recipient of many distinguished awards, including the 2004 "IEEE Chester Sall Award", HKUST "Bechtel Foundation Engineering Teaching Excellence Award" in 2004, and the "Best Paper Award" at the 2005 International Workshop on System-On-Chip for Real-Time Applications. He is a member of technical program committees of a number of international conferences including the IEEE Custom Integrated Circuit Conference CICC'2006, CICC'2007, the IEEE Consumer Electronics Conference CEC'2007, Design Automation and Test in Europe DATE2007, DATE2008. He is the general cochair of the IEEE International Symposium on electronic design test and applications, Hong Kong 2008, and the general cochair of the IEEE Conference on Biomedical Circuits and Systems, Beijing, 2009. He is also on the editorial board of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, and the *Journal of Sensors*. He is a member of IEEE CAS committee on sensory systems.



**Yan Wang** (M'10) received the B.Eng. degree from the Faculty of Science and Engineering, City University of Hong Kong, Hong Kong, in 2005, and M.Phil. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2007, where he is currently pursuing the Ph.D. degree in electronic and computer engineering.

His current research interests focus on the hardware implementation of compressive sampling on focal plane CMOS image sensor.