On-chip Focal-Plane Compression for CMOS Image Sensors

THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

by

WANG Yan

Master of Philosophy (MPhil), HKUST

This thesis is presented for the degree of

Doctor of Philosophy of

The University of Western Australia

School of Electrical, Electronic and Computer Engineering

The University of Western Australia

2012

On-chip Focal-Plane Compression for CMOS Image Sensors

by WANG Yan

The School of Electrical, Electronic and Computer Engineering

The University of Western Australia

Abstract

Miniature cameras have become an integral feature of today's networked multimedia consumer products. The ever increasing demand for low cost ultra-compact multimedia products is behind a growing effort towards integrating the different circuit components of a camera system onto a single-chip. Such an integration has become possible using microelectronics industry standard CMOS fabrication process, which enables the fabrication of a CMOS pixel array together with image processing circuitry. This thesis investigates the challenges of integrating the image compression block into a CMOS camera. The direct implementation of standard image compression algorithms like JPEG would result in prohibitively large power and silicon area budgets because image compression standards like JPEG are computationally and resource intensive.

To address this issue, this thesis introduces a number of hardware friendly image compression schemes suitable for integration with CMOS imagers. Depending on the target application, the different proposed schemes can offer different trade-offs between image quality, memory requirements, silicon and power budget.

A novel image compression processor based on predictive coding, adaptive quantization and Quadrant Tree Decomposition (QTD) algorithms featuring low complexity, low power, and high compactness was proposed and successfully implemented in CMOS $0.35\mu m$ technology. The processor was integrated with a 64×64 Time-to-First Spike (TFS) Digital Pixel Sensor (DPS) array. The pro-

cessor occupies $0.55mm^2$ silicon area and consumes 2 mW at 30 frames/s.

A second image compression scheme based on visual pattern image coding (VPIC) and optimized for TFS DPS was subsequently proposed to further improve image quality. Intensive multiplication and square root computations are replaced with addition and shift operations. Image quality with *Lena* image reported was 29 dB at 0.875 Bit-Per-Pixel (BPP).

The second part of the thesis explores potential applications of the newly introduced compressive sampling paradigm. The latter addresses the inefficiency of traditional signal processing pipeline which involves sampling followed by compression. Exploiting compressive sampling theory, we propose novel spatial and bit domain schemes that simultaneously sample and compress images with no computation. Compressed images were reconstructed using l_1 -norm minimization linear programming algorithms. Reported experimental results from the implemented FPGA platform show reconstruction quality of 29 dB at 2 BPP for 256×256 image.

Finally, a novel image compression method based on vector quantization (VQ) with shrunk codeword and reduced number of searches was proposed and implemented in FPGA. The quality of *Lena* image reported was 29.26 dB at 0.5625 BPP, with 0.57 dB sacrifice but 96.54%, 96.72%, 96.8%, and 99.47% reduction in the number of additions, subtractions, multiplications, and square roots operations, respectively, required by conventional full search VQ.

Acknowledgements

I would like to take this chance to thank my supervisor Prof. Farid Boussaid for helpful discussion, research direction steering, and contribution seeking during the course of the thesis writing.

I would like to gratefully acknowledge Prof. Amine Bermak from HKUST for his patience in co-supervision, thought-enlightening suggestions, friendship and encouragement during my research. His professional expertise in both technical knowledge and paper writing skills have helped me a lot.

I would also like to thank Prof. Abdesselam Bouzerdoun from the University of Wollongong, Australia for his technical input during this research.

I would like to thank Dr. Shoushun Chen for his active participation in this work. He provided me with valuable technical advise and support.

Thanks to all my colleagues in my Lab. Dr. Steven Zhao, Dr. Matthew Law, Dr. Milin Zhang, Dr. Xiaoxiao Zhang, Miss Rachel Pan, Mr. Frank Tang, Mr. Denis Chen, Mr. Vandy Wang, and Mr. Hassan Mohammad, who all helped with suggestions in my research work.

Thanks are also due to Mr. Luke for his technical support in chip bonding and testing and his willingness to help.

Finally, I would like to thank my family for their continuous support in this marathon.

Table of Contents

Title Page	i
Abstract	iii
Acknowledgements	v
Table of Contents	vii
List of Figures	xi
List of Tables	xv
1 Introduction	1
1.1 Digital Cameras	1
1.2 Challenges	3
1.3 Contribution of This Work	4
1.4 Thesis Organization	5
2 On-chip Compression in Solid-State Image Sensors	7
2.1 Solid State Image Sensors	7
2.1.1 CCD Image Sensor	8
2.1.2 CMOS Image Sensor	11
2.2 Review of Focal Plane Image Compression	16
2.2.1 DCT Processor	17
2.2.2 Entropy Coding Processor	18
2.2.3 HAAR Processor	19
2.2.4 Conditional Replenishment	20
2.2.5 AER Processor	21

2.2.6	Charge Sharing Based Wavelet and SPIHT coding Processor	22
2.2.7	QTD Processor for DPS	23
2.3	Performance Analysis and Comparison	24
2.4	Summary	26
3	Proposed Focal Plane Compression Imager	29
3.1	Imager Architecture	30
3.2	Pixel design and operation	31
3.3	Image Compression - Algorithmic Considerations	34
3.3.1	Predictive Boundary Adaptation	34
3.3.2	Hilbert scanning	38
3.3.3	QTD Algorithm with Pixel Storage Reuse	40
3.3.4	Simulation Results	44
3.4	VPIC Image Coding	47
3.4.1	Visual Pattern Image Coding	47
3.4.2	Isometry Operations	50
3.4.3	Simulation Results	51
3.5	VLSI implementation	53
3.6	Experimental results and Comparison	54
3.7	Summary	58
4	Compressive Sampling	61
4.1	Compressive Sampling	62
4.2	Review of CS Imaging Systems	66
4.2.1	Rice Single-Pixel Camera	66
4.2.2	Application-specific Imagers	68
4.3	Practical Challenges to CMOS CS Imagers	69
4.3.1	CMOS CS Imagers	69
4.3.2	Dynamic Range	72
4.3.3	Non-negativity and Photon Noise	73
4.3.4	Reconstruction Methods	74
5	Proposed CS Imaging Systems	77
5.1	Spatial Domain CS System	78

5.2	Bit Domain CS System	83
5.3	Hybrid System	89
5.4	Hardware Implementation and Experimental Results	91
5.4.1	FPGA Implementation of Spatial Domain CS System . . .	91
5.4.2	FPGA Implementation of Bit Domain CS System	91
5.4.3	FPGA Implementation of Hybrid System	93
5.5	Summary	95
6	Compressively Sampled Vector Quantization	97
6.1	Vector Quantization	98
6.2	CS in Vector Quantization	99
6.3	Experimental Results for the CSVQ System	104
6.4	Fast Searching Algorithm	105
6.5	Architecture	113
6.6	Performance Comparison	118
6.7	Summary	120
7	Conclusion	123
	List of Publications	127
	Bibliography	129
	Glossary	143

List of Figures

1.1	A typical digital camera system	2
1.2	CCD Camera System made by DALSA	2
1.3	The microphotograph of a single chip CMOS camera	3
2.1	Photocurrent generation in a reverse biased photodiode	9
2.2	Basic Interline Transfer CCD	10
2.3	Potential wells and timing diagram of a CCD	11
2.4	Block diagram of a CMOS image sensor	12
2.5	Basic Architecture of a Passive Pixel Sensor	13
2.6	Basic Architecture of Active Pixel Sensor	14
2.7	Basic Architecture of Digital Pixel Sensor	15
2.8	Basic Architecture of Pulse Width Modulation Digital Pixel Sensor	15
2.9	CMOS Video Camera with On-chip Compression	17
2.10	JPEG Encoder consists of MATIA and FPGA	18
2.11	Architecture of CMOS Imager with Predictive Focal Plane Compression	19
2.12	Architecture of algorithmically multiplying CMOS computational image sensor	20
2.13	System Architecture of the whole Sensor and the Compressor	24
3.1	Architecture of the overall imager including the sensor and the processor	31
3.2	Pixel Schematic and Timing Diagram	32
3.3	1-bit Adaptive Quantizer combined with DPCM.	35
3.4	Morton Z Scan and Hilbert Scan	37
3.5	Basic scanning patterns found in Hilbert scan.	39

3.6	Hilbert Scanning Patterns between Hierarchies of Quadrants . . .	40
3.7	Block Diagram of Shift Registers in QTD	41
3.8	Compression Performance as a function of η_0	43
3.10	Basic Visual Patterns for 4×4 block	49
3.11	Eight Canonical Isometries for 4×4 Image Block	50
3.12	Edge contour after image reconstruction from VPIC decoding . .	52
3.13	Microphotograph and Layout of the Chip	53
3.14	FPGA Based Test Platform	56
3.15	Captured Images from the Prototype Chip	57
4.1	Experiment in sparse signal Compressive Sampling	65
4.2	Example of Compressive Sampling for a 2D image	65
4.3	Experimental Setup of the Single-Pixel CS Camera	67
4.4	Experimental Setup for Single-Pixel CS Camera	67
4.5	Images Captured by Single-Pixel Camera	68
4.6	Implementing CS Imaging by 2-D Separable Transform	70
4.7	Random Convolution CS Imaging in CMOS	71
4.8	2D Random Scrambling CS Imaging in CMOS	72
5.1	The Proposed Image Compression System with Random Sub-sampling	81
5.2	Conversion time for the Nonuniform Time-Domain Quantizer . .	84
5.3	Time-to-First Spike Digital Pixel	86
5.4	Signal Conditioning by Noise Offset before Convex Optimization .	88
5.5	Testing Platform with FPGA and Camera	91
5.6	Sample Images from FPGA Experiments	92
5.7	Acquired images with M -MSBs and corresponding reconstructed images	94
5.8	Reconstructed images with 50% samples acquired with M -MSBs .	95
6.2	Illustration of Vector Quantization	101
6.3	Typical VQ Encoding and Decoding Process	101
6.4	Diagram illustrating CSVQ for Image Compression	103
6.5	Reconstructed CSVQ Results for <i>Lena</i> Image	106
6.6	Implementation of the Proposed Modified Prediction Scheme . . .	110

6.7	Potential Values of the Sorted Codebook	112
6.8	Architecture of the proposed CSVQ image compression scheme . .	115
6.9	The architecture of the Truncated Hadamard Transform Processor	116
6.10	Absolute Difference Accumulator and Two's Complement	116

List of Tables

3.1	Average performance of 20 test images from scientific image database under different operating modes	43
3.2	The Experimental Results of the Simplified VPIC for Image Coding in terms of Peak Signal-to-Noise Ratio	52
3.3	Comparison Results of the No. of flip-flops	54
3.4	Summary of the chip performance.	55
3.5	Comparison of our design with some other imagers with on-chip compression reported in the literature	56
5.1	Comparison of the Computational Complexity Between Different Sensing Matrices for Hardware Implementation in Digital Domain	80
5.2	Simulation Results Comparing our Proposed Matrix with the Gaussian one	82
5.3	Comparison of the Acquired Image with only M -MSBs and Its Reconstruction Using Convex Optimization	87
5.4	The PSNR of the recovered image with 50% measurements and with M -MSBs	90
5.5	The FPGA Resource Utilization of the Proposed Systems	95
6.2	Correct Matching Probability for Different Dimensions m after Compressive Sampling	102
6.3	PSNR for CSVQ for a 4×4 block when Codebook Size is 512 . .	103
6.4	Image Quality Results in terms of Peak Signal-to-Noise Ratio for <i>Lena</i> with Different No. of Measurements	105
6.5	Comparison between the Original PDS Algorithm and the Proposed PPDS Algorithm with Different Prediction Schemes	111

6.6	PSNR and the Average Number of Searches for FSVQ with 4×4 Block When Codebook Size is 512	112
6.7	Comparison of the Computational Complexity of the Three Fast Searching Schemes	114
6.8	FPGA Resource utilization for the CSVQ system	116
6.9	Performance Comparison of the 4 Compression Algorithms	118

Chapter 1

Introduction

1.1 Digital Cameras

The rapid advancement in digital camera systems has enormously contributed to the development of a variety of multimedia products. For example, digital cameras are being integrated into smart phones [1], tablet computers [2], video game remote handles [3], or vision restoration bio-sensors [4]. The system diagram of a typical digital camera is illustrated in Fig. 1.1. Light from the scene of view is focused by an optical lens. Light travels typically through a microlens array and color filter array (CFA) before impinging onto the pixel array. Incident photons are sensed through changes in voltage, current or capacitance. Acquired signals are subsequently processed to remove fixed pattern noise before being scaled by automatic gain control (AGC) (Fig. 1.1). Signals are then digitized by an analog-to-digital converter (ADC). Digital signal processing is then carried out for color processing, image enhancement and compression [5].

The core of the digital camera system is formed by the image sensor, which will fundamentally determine the quality of the output image. High-end digital cameras used in professional photography, medical or scientific imaging [6] rely mostly on charge-coupled device (CCD) image sensors. CCDs are manufactured using an optimized fabrication process for high performance photodetectors with high quantum efficiency and low noise. However, the CCD process does not allow the integration of the image sensor with other camera processing modules (Fig. 1.2). These modules need to be fabricated as separate companion chips using

semiconductor industry CMOS fabrication process. As a result, commercial CCD camera systems will be bulkier with different modules and chips.

Figure 1.1: A typical digital camera system [5]

Fabricating photodetectors using the semiconductor industry standard CMOS process enables the concept of a fully integrated camera-on-chip system (Fig. 1.3). The fully integrated product results in significantly reduced manufacturing cost, low power consumption and reduced system size, making it particularly suitable for integration into a wide range of consumer products. [7]

Figure 1.2: A CCD Camera System made by DALSA: many chips like DSP, ASICs and Memory are required besides CCD image sensor [8]

Figure 1.3: The microphotograph of a single chip CMOS camera [9]

1.2 Challenges

The ever increasing demand for compact and low-power imaging systems in mobile devices has been a driving force in the development of single-chip cameras. Such a system integrates many camera functions into a single chip. Usually, most of the image processing functions of a camera are implemented on different chips that are interfaced with the image sensor using an external processor, i.e., DSP or FPGA [5]. Image processing functions that a typical camera offers include Auto White Balance, Color Interpolation, Color Conversion and Formatting, Gamma Correction, Red Eye Removal, Sharpness Adjustment, and Image Compression, to name a few [7].

However, image compression is an important image processing function that is still performed using external processors. This is because the widely adopted industry standard JPEG image compression algorithm has a high level of complexity, resulting in turn in high power consumption.

Computationally intensive algorithms require a prohibitively high power budget but the area required for data storage could be small. Less computationally intensive algorithms are less power hungry but resulting compressed data require more memory to be stored. In selecting the image compression scheme to be implemented on-chip, one needs to consider the trade-off between image quality, compression ratio, and power consumption/silicon area budgets. To address this,

many researchers have been investigating alternative image compression schemes more suitable to low power compact VLSI implementations [10, 11]. Examples include Wavelet based algorithm [12] or Golomb-Rice entropy coder [13].

1.3 Contribution of This Work

The aim of this thesis is to investigate hardware friendly compression schemes suitable for integration with the CMOS image sensor. We propose novel image compression schemes based on the universally adopted “sample and then compress” paradigm. We also explore potential applications of the newly introduced “Compressive Sampling” paradigm, which enables compression during image acquisition. Each of the proposed image compression schemes offer different trade-offs between image quality, memory requirement, silicon area and power budgets. The contribution of this thesis can be summarized as follows:

1. A 64×64 time domain digital pixel sensor array integrated with a robust and compact image compression processor was successfully implemented in CMOS $0.35\mu m$ technology. The chip occupies a silicon area of $3.2 \times 3.0 mm^2$. It consumes 17 mW at 30 frames/s. Simulation and measurements results show compression ratio at 0.75 Bit-per-Pixel (BPP), while maintaining reasonable PSNR levels. The sensor is equipped with non-destructive storage capability using 8-bit Static-RAM device embedded at the pixel level. The pixel readout follows Hilbert scanning path to exploit inter-pixel redundancy. The image compression processor performs adaptive quantization based on Fast Boundary Adaptation Rule (FBAR) and Differential Pulse Code Modulation (DPCM) followed by an on-line, least storage Quadrant Tree Decomposition (QTD) processing.
2. To further improve image quality, a novel image compression scheme based on visual pattern image coding (VPIC) [59] was proposed for Time-to-First Spike (TFS) Digital Pixel Sensor (DPS). The computational complexity in calculating block mean, gradient magnitude was significantly reduced by replacing multiplication and square root operations with addition and shifting operations. The quality of the reconstructed image measured in terms of

PSNR for *Lena* image at 0.875 BPP was around 29 dB. Each image block was compared with visual patterns that exploit psychovisual redundancy. The closest pattern was found using binary correlation operations. Isometry operations were proposed to re-utilize each of these visual patterns. This has eliminated the computation for gradient angle and edge polarity by on-line pattern expansion and exhaustive search.

3. A novel image compression scheme based on Compressive Sampling (CS) was proposed and implemented on an FPGA platform interfaced with a CMOS imager. The scheme compresses image in both spatial and bit domains. Results show that for 256×256 resolution images, a PSNR of 29 dB is achieved at 2 BPP. The proposed sensing matrix enables multiplication-free CS encoding and reduces the integration time of the TFS DPS. This leads to low-complexity encoding as no computation is required at the front-end. Compressed data were reconstructed by a convex optimization technique making use of the robust recovery property of CS theory.
4. An image compression scheme that integrates CS into Vector Quantization (VQ) was proposed to reduce the number of Euclidean Distance computations by roughly $1/2$ compared to the traditional VQ scheme. A predictive partial distance search (PPDS) algorithm for fast codebook searching was also proposed to further boost the speed in VQ encoding. When the number of measurements is $m = 9$ and for a 4×4 image block, the PSNR sacrifice was 0.57 dB but the number of additions, subtractions, multiplications, and square roots operations were only 3.46%, 3.28%, 3.2%, and 0.53%, respectively, of that of a conventional full VQ search. The proposed scheme was validated on an FPGA platform. It is well suited to wireless sensor network applications (e.g. environment monitoring) for which scenes are captured at low frequencies.

1.4 Thesis Organization

The organization of this thesis is as follows. Chapter 2 reviews solid-state image sensors together with prior works on focal plane image compression for CMOS

image sensors. Chapter 3 presents the architecture, operation and VLSI implementation of a CMOS image sensor integrating image capture, storage and compression on a single chip. Chapter 4 introduces the Compressive Sampling (CS) theory, its applications in various imaging systems, and the practical challenges for its integration into CMOS image sensors. Chapter 5 presents spatial domain, bit domain and hybrid mode image compression schemes based on the CS paradigm. An FPGA platform implementation is described to validate results. Chapter 6 explores the application of CS framework into the block-based image compression algorithm Vector Quantization (VQ) in order to reduce the number of Euclidean Distance computations required in the codebook search. An FPGA hardware architecture is also presented. Chapter 7 concludes this work and discusses future work.

Chapter 2

On-chip Compression in Solid-State Image Sensors

2.1 Solid State Image Sensors

Digital cameras capture images using solid state image sensors with resolutions ranging from 320×240 for lowest quality video recording to 10560×10560 for astrometry applications. Each pixel comprises a photodetector together with circuitry to read-out and/or process sensed electrical signal. Pixel size ranges from $1.43\mu m \times 1.43\mu m$ [14] in consumer digital camera to $24\mu m \times 24\mu m$ image sensor with extremely high quantum efficiency [15].

The solid state photodetector generates a photocurrent, whose magnitude is proportional to the intensity of the incident optical power. The most widely used photodetectors are the photodiode and the photogate. The photodiode is essentially a reverse-biased p-n junction while the photogate is a MOS capacitor [16]. The photocurrent generated in reverse-biased photodiode is mainly attributed to the current generated in the depletion region (space-charge region) and is partially attributed to the currents generated in the n-type and p-type quasi-neutral regions, as shown in Fig. 2.1. Incoming photons create electron-hole pairs in the depletion region i_{ph}^d , holes in the n-type quasi-neutral region i_{ph}^p , and electrons in the p-type i_{ph}^n . Therefore, the total current is

$$i_{ph} = i_{ph}^d + i_{ph}^p + i_{ph}^n \quad (2.1)$$

Ideally, each incident photon would generate one electron-hole pair. However, photo-generated electrons and holes could recombine. When this happens, no contribution to the photocurrent is made by the incident photon. The Quantum Efficiency (QE) is the percentage of incident photons that effectively generate electron-hole pairs. The photodiode sensitivity is expressed by the spectral response $SR(\lambda)$, which is directly related to QE [16]. The relationship between $SR(\lambda)$ and QE is shown here:

$$SR(\lambda) = \frac{q}{hc/\lambda} \cdot QE = \frac{QE \cdot \lambda [\mu m]}{1.23} \quad (2.2)$$

The dark current i_{dc} of a photodiode is the main factor that limits its signal swing and thus dynamic range because a photocurrent that is considerably smaller than the dark current cannot be easily measured. As the name suggest, dark current is the photocurrent measured under no illumination. It is due to the recombination of carriers in the depletion region but also to leakage associated with defects in the silicon.

Another limitation is the shot noise which is induced by random fluctuations of the photocurrent flows with discrete charges. It is expressed as:

$$i = \sqrt{2qI\Delta f} \quad (2.3)$$

where q is the charge of an electron, I is DC diode current, and Δf is the bandwidth of the diode.

Photocurrent is usually in the range of tens to hundreds pA . As a result, a period of exposure time has to be allocated for charges to be accumulated before taking any measurement. Prior to sensing, a photodiode is first reset to a voltage in a reverse-biased mode. It is then exposed to light, with the generated photocurrent discharging the photodiode parasitic capacitance. After the exposure period (integration period), the charge or voltage is then readout.

2.1.1 CCD Image Sensor

Charge-coupled device (CCD) was invented in 1969 at AT&T Bell Labs. It was firstly developed as a shift register and has now become one of the most widely

Figure 2.1: Photocurrent generation in a reverse biased photodiode [17]

used solid state image sensors in digital cameras. The architecture of a CCD image sensor consists of an array of photogates (MOS capacitors) [18]. Each capacitor accumulates electric charges proportional to the intensity of incident light. After exposure, the charges on the capacitor are typically transferred out with a 3-phase clock control, which serially shifts each capacitor's charges to its neighbor.

Fig. 2.2 depicts the block diagram of the widely used interline transfer CCD image sensors. MOS capacitors are closely spaced to increase quantum efficiency. A row of charges are firstly shifted out simultaneously through the vertical CCDs. The charges in each horizontal CCD are serially shifted out and fed onto a charge amplifier that outputs a voltage. Assume the resolution of the CCD image sensor is $l \times h$, then the clock speed for the horizontal CCD will be h times that of the vertical CCD. The voltage generated by the charge amplifier is then sampled, quantized, and further processed.

The operation principle of a 3-phase clocked CCD is illustrated in Fig. 2.3 in which ϕ_1 , ϕ_2 and ϕ_3 are three clocks [19]. When clock ϕ_1 is high, a potential well is generated beneath the electrode in the p-substrate. The charges (electrons) are concentrated in the potential well. When clock ϕ_2 is driven to high, a potential well underneath its electrode is formed. The latter is connected with the potential well of the clock ϕ_1 . The electrons are then shared by the two potential wells through repulsive forces between electrons. The clock ϕ_1 then falls to zero

Figure 2.2: Basic Interline Transfer CCD

gradually, which causes the potential well to diminish gradually. Finally, almost all electrons are shifted to the potential well of ϕ_2 . Thermal diffusion, fringing fields and lateral drift of electrons are the main factors for leakage through charge transfer in CCDs. By applying clock voltage to gate electrodes in correct order, charges could be properly transferred.

An $l \times h$ CCD image sensor requires up to $l + h$ transfers. The clock needs to be slow enough so as to maintain high transfer efficiency, but also fast enough so as to reduce leakage. The limitation in readout speed/frame rate is a big obstacle to high speed applications. A commercial 6096×4560 CCD image sensor with pixel size $7.2\mu m \times 7.2\mu m$ can boast 0.8 fps by interline transfer given a 0.999999 transfer efficiency [20].

The charge transfer efficiency should be as close to 100% as possible to ensure image quality. This explains the costly specialized and optimized CCD fabrication process.

CCDs can offer high quality images with low noise, low dark current, high uniformity and high quantum efficiency, making it very suitable for applications like medical imaging, aerospace & defense imaging, industrial imaging, and professional cameras. However, CCDs consume significant power because of their high supply voltage. Usually its positive power supply is $+15V$ and its negative power supply is $-8V$, with the sensor array continuously operating at high frequency.

Another disadvantage of CCD camera is that the CCD image sensor cannot

Figure 2.3: Potential wells and timing diagram during the transfer of charge in a three-phase CCD [19]

be integrated with CMOS based circuits and systems, such as the ADCs, or image enhancement processors. These functions are all implemented in off-sensor companion camera chips.

Recent advances in CCD image sensor technology have further increased quantum efficiency and decreased dark current, pixel size, and operating voltage. Companion circuits are also becoming more and more integrated, making the camera system smaller and more power efficient [21]. CCD cameras are more geared towards high performance consumer imaging products. The lower cost imaging segment is currently being dominated by CMOS Image Sensor (CIS) technology [22].

2.1.2 CMOS Image Sensor

Complementary Metal Oxide Semiconductor (CMOS) imaging technology was developed in the 1960's. In the beginning, it exhibited low image quality because

of the limitations of older CMOS technology. In the 1990's, with the improvement in lithography and fabrication process, CMOS image sensor technology surfaced to become the technology of choice for lower cost imaging products.

CMOS image sensors are fabricated using standard CMOS processes with no or minor modifications. Fig. 2.4 shows a typical CMOS image sensor architecture. Each pixel in the array is addressed through a horizontal word line and the charge/voltage signal is read out through a vertical bit line. The readout is done by transferring one row at a time to the column storage capacitors, then reading out the row using the column decoders and multiplexers. This readout method is similar to that of a memory structure.

Figure 2.4: Block diagram of a CMOS image sensor [17]

The rapid emergence of CMOS image sensor technology is the result of the following factors [21]:

1. With improved lithography and process control of CMOS fabrication, image quality of CMOS sensor could potentially challenge CCDs
2. Camera on chip could be achieved by integrating companion modules on the same chip as the image sensor, thus, the size of the complete imaging system could be reduced as a result of integration
3. Power consumption could be significantly reduced due to lower power supply level used in CMOS circuits

4. CMOS production lines could be used to fabricate both imagers and digital/memory circuits
5. Fabless CMOS design companies can design their camera modules and then hand them over to their foundry of choice

There are a number of pixel topologies proposed for CMOS image sensors.

2.1.2.1 CMOS Passive Pixel Sensor

In 1960's, CMOS Passive Pixel Sensor (PPS) was invented (Fig. 2.5). In a passive pixel [23], photodiode converts photons into electrical charge. The charges in the pixel array are then readout row by row. The global charge-to-voltage amplifier then converts the charges into an analog voltage. The advantage of PPS is its small pixel size and large fill factor. When large resolution is required, the parasitic capacitance of the column bus increases proportionally with the resolution. When the latter becomes significantly larger than each photodiode's capacitance, the noise issue becomes problematic during the pixel readout. Another disadvantage of PPS is that the pixel readout is destructive. The architecture of PPS is illustrated in Fig. 2.5. Note that a 'Column Amplifier' is shared by all elements of the column. The 'Col Select' signal serially reads out the value in each 'Column Amplifier' to the 'Output Amplifier'.

Figure 2.5: Basic Architecture of a Passive Pixel Sensor (PPS) [17]

2.1.2.2 CMOS Active Pixel Sensor

In 1993, the first CMOS Active Pixel Sensor (APS) [24] was demonstrated at a resolution of 28×28 pixel. In contrast to PPS, an APS integrates a charge-to-voltage amplifier into each pixel (Fig. 2.6). The amplifier isolates the photodiode from the data bus, making the pixel readout process non-destructive.

The typical 3-Transistor APS is illustrated in Fig. 2.6. The photodiode is firstly reset by turning on the transistor M_{rst} . All its integrated charges are then cleared when connecting it to the power supply V_{rst} . It is then turned off to allow the pixel to integrate for a predefined time period. After charge integration, the analog buffer (source follower) transistor M_{sf} reads out the pixel voltage level without destructing the accumulated charges on the photodiode's parasitic capacitance. When the row select transistor M_{sel} is turned on, a row of the pixel array is then readout by an external circuitry.

Most of today's CMOS image sensors adopt a 4T CMOS APS architecture, which has an extra transfer gate transistor to control the pinned photodiode. Extra transistors have also been introduced to perform additional functions such as global shutter and correlated double sampling (CDS).

Figure 2.6: Basic Architecture of Active Pixel Sensor (APS) [16]

2.1.2.3 CMOS Digital Pixel Sensor

In 1994, the first CMOS Digital Pixel Sensor(DPS) [25] with pixel-level Analog-to-Digital Converter (ADC) was proposed. In this work, 1-bit Delta Sigma modulator was implemented in each pixel with the Decimation Filter implemented off-chip. The work illustrated how a digital representation of the signal could be stored at the pixel level. The technology node was $1.2\mu\text{m}$ CMOS technology, with a pixel size of $60\mu\text{m} \times 60\mu\text{m}$, which is quite large. The reason for this is the capacitor **C1** used in the integrator of the delta sigma modulator (Fig. 2.7). **C1** should normally be very large, that is of order of dozens of femto farads.

Figure 2.7: Basic Architecture of Digital Pixel Sensor (DPS) (Adapted from [25])

Figure 2.8: Basic Architecture of Pulse Width Modulation (PWM) Digital Pixel Sensor (DPS) (Adapted from [26])

A DPS architecture without any pixel-level ADC has also been explored. Pulse

Width Modulated (PWM) Digital Pixel Sensors were presented in [26, 27]. In this pixel architecture, an 8-bit SRAM is embedded into each pixel. The global integration time is non-linearly converted into pixel intensity using digital control circuitry before being fed to the global data bus. The pixel intensity is represented in a gray code format to minimize the power consumption and crosstalk in the data bus. The nonlinear conversion from timing information to pixel intensity is achieved using a look-up table. With device scaling, the area associated to in-pixel digital circuits such as comparator, buffer, SR Latch, and SRAM has shrunk, yielding better fill factor. In contrast, for APS, device scaling makes the analog circuits' performance degrade because noise sources do not scale down with power supplies.

2.2 Review of Focal Plane Image Compression

With rapid advances in network and multimedia technology, real time image acquisition and processing has become challenging because of ever increasing image 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 [28], such as ZigBee (up to 250 kbps) and Bluetooth (up to 1 Mbps). Even at the data rate of Bluetooth, a conventional image sensor could barely stream an uncompressed 320×240 8-bit video at 2 frame/s ($\frac{1 \times 10^6 \text{ bit/s}}{320 \times 240 \times 8 \text{ bit/frame}} = 1.6 \text{ frame/s} < 2 \text{ frame/s}$). **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 cosine transform (DCT) and wavelet transform (WT), among various block transforms, are popular in image/video compression standards such as JPEG, MPEG, H.261 and H.263. However, implementation of these standards in cameras is computationally expensive, requiring a dedicated digital image processor in addition to the image sensor [10]. 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, sys-

tem 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 [12] [29] [30] [31] [32] [33] [34] [47] [54] [55].

2.2.1 DCT Processor

In [29], an 8×8 point analog 2D-DCT processor implemented using switched capacitor circuits is reported in Fig. 2.9. Block based readout was used with the

Figure 2.9: CMOS Video Camera with On-chip Compression (Adapted from [29])

pixel sensor array divided into blocks of 8×8 size. Each block is readout to perform the 2D-DCT and then digitized by an Analog-to-Digital Converter/Quantizer (ADC/Q). The 2D-DCT is performed using an analog 1D-DCT processor. Each column of a block goes through the 1D-DCT and the intermediate values are stored in the analog memory. The transpose of the coefficients in the analog memory is then computed before feeding it to the 1D-DCT processor to obtain the final 2D-DCT coefficients. These coefficients are then digitized by the ADC/Q with a simplified quantization table, in which each element is a power of 2. The Variable Length Coding (VLC) processor is implemented off-chip to further reduce the data transmission bit rate.

In [32], floating gate technology is used to store the DCT matrix and compute the 2D-DCT coefficients. Floating gate is basically polysilicon gate encircled by silicon dioxide. The charges on the floating gate could be adjusted by the gate

voltage in the program mode. In the transform (operation) mode, the drains of floating gates in the same row are connected with current-to-voltage (I-V) converters to transform currents into bias voltages for matrix-vector multiplication. Current-mode differential vector matrix multiplier (VMM) is used for matrix-vector multiplication. Besides DCT, other transforms such as discrete sine transform (DST), Haar wavelet transform, and Walsh-Hadamard transform have also been implemented [32]. The transformation is done on-chip, while the encoding process for JPEG standard is performed off-chip in an FPGA. The JPEG encoder that consists of Matrix Transform Imager Architecture (MATIA) and Field Programmable Gate Array (FPGA) is illustrated in Fig. 2.10.

Figure 2.10: JPEG Encoder consists of MATIA and FPGA (Adapted from [32])

2.2.2 Entropy Coding Processor

However, the aforementioned designs do not actually implement compression on the focal plane since the entropy coding stage is located off-chip to minimize chip size and cost. In [33], a 44×80 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 150mW and a size of $2.596mm \times 5.958mm$ in $0.35\mu m$ CMOS technology. The chip architecture is illustrated in Fig. 2.11. The median predictor is used with the analog pixel value X and predicted value $\hat{X}$ digitized by the column-level A/D converter. The Golomb-Rice entropy coder is integrated into the column processor to compress the image data.

Figure 2.11: The architecture of CMOS Imager with Predictive Focal Plane Compression (Adapted from [33])

2.2.3 HAAR Processor

In [12, 34], 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 128×128 pixel array with a die area of $4.4mm \times 2.9mm$ and a total power consumption of 26.2mW [34]. The overall imager architecture is illustrated in Fig. 2.12. The in-pixel dual frame memory is used for multiple sampling in both spatial and temporal image processing. The processing circuit consists of sign unit, binary-analog multiplier, accumulator and multiplying analog to digital converter (MADC). The switch matrix sets the transformation matrix kernel size, routes kernel coefficients and their corresponding sign bit and sends them to the sign unit and the binary analog multiplier. The MADC is used to multiply the pixel value with its corresponding coefficient for convolutional transform. The Haar transform was used to validate

functionality of the processor.

Figure 2.12: The architecture of algorithmically multiplying CMOS computational image sensor (Adapted from [34])

2.2.4 Conditional Replenishment

A 32×32 image sensor exploiting conditional replenishment compression scheme was reported in [35]. This compression technique removes the inter-frame temporal redundancy by subtracting pixel values in consecutive frames. This is achieved by storing the pixel value of current and previous frames in two on-chip capacitors. When the difference between the two is larger than a threshold, the Flag for this pixel is set to 1 and its value as well its address is streamed out. A vertical shift register scans the image sensor row by row, where an horizontal shift register shifts out only pixels with a flag set to 1. To maintain a fixed data rate the threshold is controlled with respect to the number of activated pixels. The compression ratio of 10 : 1 was achieved with no significant degradation to low motion activity scenes. When fairly large moving area appears in a scene, a compression ratio of 100 : 15 was used. By adjusting the frame rate of the image sensor, the motion change in each frame can be varied: higher frame rate leads to less motion changes. The sensor was targeted at the application requiring more

than 1000 frames/s.

The next generation prototype [36] uses a 32×32 pixel sensor array with the conditional replenishment scheme implemented using a column parallel approach. Pixel values for the previous frame are stored in a separate capacitor array. Comparators were moved out from individual pixels to form a row. Each one was shared by a column of pixels. Its power consumption is reduced and fill factor for each pixel is increased, but its speed is sacrificed. Smart scanning and rate control were also implemented, operating between 30 and 1200 frames/s with fixed rate for activated pixels.

In another implementation [37], a 64×64 image sensor with a compression algorithm exploiting correlation between two consecutive frames was proposed. The difference between two consecutive frames is coded, note that most difference data will be centered around zero. n is the controllable compression parameter, when absolute difference value is smaller than 2^{n-1} , $2^n + 1$ bits are allocated for representation, when it is larger, then 10-bit is allocated. The smart shift register is implemented to shift out the compressed data and corresponding address. The sensor is suitable for high frame rate scenario, i.e., 1000 frames/s.

2.2.5 AER Processor

In [38, 39] an Address-Event Representation (AER) processor was designed and integrated with a 128×128 CMOS vision sensor. The pixel comprises an active continuous time logarithmic photoreceptor, a self-timed switched-cap based differential amplifier together with an asynchronous communication circuit. Each pixel continuously monitors intensity changes and generate spikes as address-events when the relative changes in pixel intensity is larger than a threshold. The generated address-event is processed by handshake logic and arbiter before it is transmitted out. This AER vision sensor boasts 120 dB dynamic range and a power consumption of 23 mW.

In [40], a 90×90 AER image sensor array with in-pixel differential and comparison circuitry was reported. Its pixel monitors absolute change in illumination. This sensor is synchronous because it stores the address-event in a FIFO. Pixel size was reduced by using NMOS transistors. However, the sensor suffers from

the drawback of low dynamic range. Another inherent disadvantage is that it uses an absolute illumination change threshold, which is only useful when the scene is uniform. This sensor consumes 4.2 mW at 30 fps and produces a 20-fold compression with 51 dB dynamic range.

In [41], a QVGA (304×240) resolution AER image sensor array was reported. Each pixel is fully autonomous and comprising an illumination change detector and a time-domain photo-measurement circuitry. The change detector is a logarithmic photoreceptor operating continuously with asynchronous handshaking and arbiter circuit. This is essentially the same sensor as in [39] but with a smaller photodiode (PD). The photo-measurement is based on a time-to-first spike pulse-width-modulation (PWM) technique. The QVGA sensor consumes 50 mW in static mode and 175 mW in high activity mode. It achieves 143 dB dynamic range in static mode and 125 dB at 30 fps.

2.2.6 Charge Sharing Based Wavelet and SPIHT coding Processor

A charge-based circuit for focal plane compression chip was presented in [42]. A charge-based predictor was designed with three neighboring pixels. The Northern, Western, and Northwestern neighbors are selected to calculate the prediction by charge injection. The difference between the predicted value and the actual value, which is called the prediction error, is adaptively quantized to suppress noise in the analog circuit. The charge-based approach features a compact prediction circuit with 5 transistors, 2 capacitors, and a photodiode.

Based on this work, the focal plane decomposition [43] enabled by charge sharing is presented. One-level of decomposition was implemented which yields 2-3 dB lower PSNR than the ‘9/7’ wavelet transform after 0.25 BPP. Using this charge-based approach, the memory for buffering the interim wavelet transform coefficients can be eliminated and parallel computing in the image acquisition stage could be achieved. This work was further improved in [44] to address the charge injection problem. A new pixel topology was proposed so that the pixel value can be reused for computing prediction residual instead of being regained through photodiode reintegration.

A 3-level decomposition by charge-based prediction was designed and reported in [45]. Compared with previous works, one extra capacitor was added into this design. Charge sharing between two capacitors in the same pixel was exploited rather than between two pixels in the previous design. This approach reduces parasitic couplings. This new pixel outperforms the design in [44] by 10dB when working in 1-level decomposition mode. Charge injection is not an issue in this pixel as the injection error is very small. A processor that implements both 1-level and 3-level decomposition was reported in [55]. It consumes 0.25 mW at 30 frames/s. The charge injection error compensated for using a new correlated double sampling (CDS) architecture was proposed and successfully implemented.

A modified SPIHT focal plane compression processor prototyped in $0.35\mu m$ technology was reported in [46]. The descendant assignment and tree lists update procedures were redesigned. The obtained PSNR is 0.8 dB lower than the standard SPIHT when the bit rate is less than 1 BPP and 1 dB lower when the bit rate is more than 1 BPP.

2.2.7 QTD Processor for DPS

In [47], 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 [47]. 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. 6.3mW) and occupies a silicon area of $1.8mm^2$. The compression sensor permits to compress the data to $0.6 \sim 1$ BPP (Bit-per-Pixel). The imager uses a Morton(Z) 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. The architecture of the whole focal plane image sensor and the two building blocks of compression (the Quadrant Tree Decomposition (QTD) processor and adaptive quantizer) are illustrated in Fig. 2.13.

Figure 2.13: (A) Focal Plane Image Sensor with Successive Compression Processors; (B) Quadrant Tree Decomposition (QTD) Processor; and (C) Adaptive quantizer. (Adapted from [47])

2.3 Performance Analysis and Comparison

A performance comparison of the aforementioned prior works on focal plane image compression is given in Table 2.1. [29] reported an off-array 2D-DCT processor integrated in a CMOS image sensor. However, the quantization and entropy coding processors, which constitute the most important building blocks for image compression, were not implemented [29]. The pixel used was a 3T APS, with a fill factor of (56.6%). This was the highest among all designs in this Table. The 2D-DCT processor was designed using analog circuitry, resulting the second lowest power consumption (5.4 mW).

For the case of [32], both the DCT and Quantization were implemented on-chip in an off-array processor, but no entropy coder was implemented. The pixel contains only 2 transistors in a differential pair topology to facilitate current voltage matrix multiplication with the DCT coefficients stored in the Floating-Gates.

The fill factor (46%) was the second largest while the power consumption dramatically outperforms other designs down to $80\mu\text{W}/\text{frame}$. The main drawback of this design is inherent to the use of floating gate technology, which requires high voltage (16V) to program the coefficients.

In [34], Haar Transform was implemented both at the pixel-level but also off-array. However, the entropy coding part was not implemented. The block-matrix multiplication and convolution transformations were performed on the focal plane by programmable 8×8 digital kernel. The frame differencing were also performed on the focal plane by compact and scalable circuits. This design achieved the smallest area compared with those using the same $0.35\mu\text{m}$ technology.

In [55], the Wavelet decomposition was implemented at pixel level, but the SPIHT encoding process was carried out in the back-end processor. This decomposition was prediction based and involved intensive charge sharing process. This required large capacitors in each pixel, therefore it had the largest pixel area when comparing with others. Due to the charge sharing nature in data processing, the actual loss could be small thus low power (0.25 mW) could be possible at low frame rate (30 fps). The overall reconstructed image quality was the highest among all designs due to the fact that the Wavelet transform intrinsically outperforms DCT in image decomposition and representation.

The aforementioned works all implemented the computational intensive part of the transform-based compression algorithms, but left the encoding process to external processors. The following works all implemented compression engines and coding processors, with no external circuitry needed.

[33] implemented both image compression by predictive coding and the off-array Golomb-Rice entropy coding processor. Lossless compression was achieved at the compression ratio of $1.3 \sim 1.5$. The main drawback of this design is its higher power consumption (150 mW).

[37] implemented pixel level conditional replenishment algorithm as well as the off-array difference encoding processor. A high frame rate was required to control the level of sparseness between two consecutive frames to ensure proper data compression. The drawback of this design lies in the fact that they need to operate at higher frequencies. Otherwise, the number of activated pixels will be

too high, making compression difficult to achieve.

[41] implemented AER based compression algorithm that exploits temporal redundancy. The designed Time-to-First Spike Digital Pixel Sensor achieves the highest dynamic range (125~143 dB) with a compression scheme dedicated to moving pictures, with no compression for still images.

A similar pixel architecture (DPS) was adopted in [47], where a 6-T SRAM was embedded into the pixel. A 1-bit adaptive quantizer (FABR) was implemented before the data stream compression performed by QTD processor. The pixel area was large compared with other DPS [41]. This is mainly due to the embedded in-pixel memory and the use of an older CMOS technology ($0.35\mu m$ v.s. $0.18\mu m$). This chip consumes 20 mW with the compression processor only dissipating 6.3 mW [47]. This design exhibits 100 dB dynamic range, which is the second highest among all those reported designs. Its compression ratio was about 10 but the quality of the reconstructed image quality was not high. However, it would be suitable for target applications such as wireless sensor networks, where the environment monitoring does not require high image quality. This is because it achieves the best compromise between power consumption, dynamic range, compression ratio and image quality. Further, reduced silicon area and power consumption together with enhanced image quality could be required features for next generation prototypes.

2.4 Summary

In this chapter, we introduced the CCD and CMOS solid state image sensors with their underlying physics and operating principles. The PPS, APS and DPS CMOS image sensors were also presented. Various kinds of image compression processors were then discussed in detail. These include: 2D DCT Processors that are implemented in both analog circuit and in floating-gate, Entropy Coding Processor that executes column-level processing, Haar Wavelet Transform Processor that is implemented in mixed analog and digital mode, Conditional Replenishment Processor and AER Processor that exploit temporal difference; SPIHT Coding Processor implemented with charge sharing; and QTD Processor that is implemented in DPS. A thorough performance analysis and comparison

Table 2.1: Performance Comparison of the on-chip Focal Plane Compression Schemes

	[29]	[32]	[33]	[34]
Performance of the Overall System				
Year	1997	2006	2007	2009
Power Supply	3V	3.3V	3.3V	3.3V
Resolution	128×128	104×128	80×44	128×128
Technology	N/A	$0.5\mu m$	$0.35\mu m$ 2P4M	$0.35\mu m$
Overall Silicon Area	$5.4 \times 4.3mm^2$	$2.99mm^2$ (Pixel Array)	$2.6 \times 6.0mm^2$	$4.4 \times 2.9mm^2$
Power Consumption	5.4 mW (DCT Processor)	$80\mu W/frame$	150 mW	26.2 mW
Performance of the Sensor Array				
Sensor Type	APS	APS	APS	APS
Pixel Size	$16.1 \times 16.1\mu m^2$	$13.5 \times 13.5\mu m^2$	$34 \times 30\mu m^2$	$15.4 \times 15.4\mu m^2$
# of Transistors/Pixel	3	2	8	N/A
# of Capacitors/Pixel	0	0	1	2
Fill Factor	56.6%	46%	18%	28%
Dynamic Range	N/A	N/A	48 dB	N/A
Dark Current	$503pA/pixel$	$14.9nA/cm^2$	N/A	$36fA/pixel$
Performance of the Compression Processor				
Compression Algorithm	2D-DCT in Analog Domain	2D-DCT and Quantization	Golomb-Rice Entropy Coding	Haar Wavelet Transform
Processor Implementation	Off-array Processor	Off-array Processor	Off-array Processor	Pixel-Level Logic & Off-array Processor
Frame Rate	N/A	25 fps	N/A	30 fps
Compression Ratio	5	N/A	1.3~1.5	3.85
PSNR	35 dB	N/A	Lossless	32 dB
	[37]	[41]	[55]	[47]
Performance of the Overall System				
Year	2004	2010	2008	2007
Power Supply	N/A	3.3V	N/A	3.3V
Resolution	64×64	304×240	33×25	64×64
Technology	$0.6\mu m$ 2P3M	$0.18\mu m$ 1P6M	$0.5\mu m$ 2P3M	$0.35\mu m$ 1P5M
Overall Silicon Area	$3.05 \times 9.67mm^2$	$9.9 \times 8.2mm^2$	N/A	$3.8 \times 4.5mm^2$
Power Consumption	N/A	175 mW	0.25 mW	20 mW
Performance of the Sensor Array				
Sensor Type	APS	DPS	APS	DPS
Pixel Size	$30 \times 30\mu m^2$	$30 \times 30\mu m^2$	$69 \times 69\mu m^2$	$45 \times 45\mu m^2$
# of Transistors/Pixel	N/A	77	14	38
# of Capacitors/Pixel	N/A	2	4	0
Fill Factor	21.4%	30%	20.7%	18%
Dynamic Range	N/A	125~143 dB	N/A	> 100 dB
Dark Current	N/A	$1.6nA/cm^2$	N/A	N/A
Performance of the Compression Processor				
Compression Algorithm	Conditional Replenishment	AER Temporal Compression	Prediction Based Decomposition for SPIHT	FABR based QTD
Processor Implementation	Pixel-Level & Off-array Processor	Pixel-Level Processor	Pixel-Level Processor	Pixel-Level & Off-array Processor
Frame Rate	1000 fps	30 fps	30 fps	N/A
Compression Ratio	1.7 ~13	20~400	N/A	> 10
PSNR	~37 dB	Lossless	38dB	~23 dB

Chapter 3

Proposed Focal Plane Compression Imager

As reported in Section 2.2, most prior implementations of on-chip focal plane compression implemented either simplified schemes of a complex image compression algorithm or some building blocks of a compression standard. This chapter presents the architecture, operation and VLSI implementation of a second generation CMOS image sensor boasting image capture, storage and compression, all integrated on a single chip.

The second generation prototype integrates the following features: a) new Hilbert scanning technique and its VLSI implementation to avoid spatial discontinuities in the block-based scanning strategy; b) a 1-bit Fast Boundary Adaptation Rule (FBAR) algorithm performed on the predictive error rather than the pixel itself using Differential Pulse Code Modulation (DPCM) for improved performance; c) introduction of memory reuse technique enabling over a three-fold reduction in silicon area and d) improved pixel structure for the Digital Pixel Sensor (DPS) sensor.

The remainder of this chapter is organized as follows. Section 3.1 describes the overall imager architecture and Section 3.2 introduces the adopted digital Time-to-First Spike (TFS) image sensor. Section 3.3 discusses the algorithmic considerations for the proposed FBAR algorithm, predictive coding technique, and design strategies used for implementing the Hilbert scan as well as the Quadrant Tree Decomposition (QTD) processing involving the memory reuse concept.

Section 3.4 presents the compression algorithm based on the Visual Pattern Image Coding (VPIC) technique and its FPGA validation results. Section 3.5 discusses the VLSI implementation of the compression imager. Section 3.6 reports the experimental results and provides a comparison with other compression processors. Section 3.7 concludes this work.

3.1 Imager Architecture

Fig. 3.1 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 64×64 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 non-linearity **by adjusting the quantization times using a non-linear clock signal which is fed to the counter [48]**. Moreover, proper adjustment of the quantization timing stamps stored in a 16×256 -bit on-chip SRAM memory enables to implement various transfer functions including a log-response [49].

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 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.

Figure 3.1: Architecture of the overall imager including the sensor and the processor

3.2 Pixel design and operation

The proposed system integrates the image sensor with pixel level ADC and frame storage together with the array-based stand-alone compression processor. The sensor array adopts a time domain digital pixel sensor (DPS) [48], in which the image is captured and locally stored at the pixel level. The image array consists of 64×64 digital pixel sensors. **Fig. 3.2(a) illustrates the circuit diagram of the pixel, which includes 4 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. 3.2(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

(a)

(b)

Figure 3.2: (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.

of the sensing node VN is first reset to V_{dd} . 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_{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 [48] 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 [48]. 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.

3.3 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.

3.3.1 Predictive Boundary Adaptation

The proposed boundary adaptation scheme can be best described using an ordered set of boundary points (*BP*) $y_0 < y_1 < \dots < y_{i-1} < y_i < \dots < y_{N-1} < y_N$ delimiting N disjoint quantization intervals $R_1, \dots, R_i, \dots, R_N$, with $R_i = [y_{i-1}, y_i]$. The quantization process is a mapping from a scalar-valued signal x into one of 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 can

therefore be defined as:

$$d(x, Q(x)) D_r \equiv \sum_{i=1}^N |x - y_i|^r p(x) dx \quad (3.1)$$

It has been shown that using Fast Boundary Adaptation Rule (FBAR) [50] 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 $D_r(i) = D_r/N$ [50]. 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 η . 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**

Figure 3.3: 1-bit Adaptive Quantizer combined with DPCM.

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. 3.3. 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:

$$BP_P = Reg0 \times 1.375 - Reg1 \times 0.75 + Reg2 \times 0.375 \quad (3.2)$$

The coefficients 1.375, -0.75, and 0.375 could be implemented by only shift and add operations. They are the approximated values that are obtained through extensive simulation. Compared to the scheme reported in [47], 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:

$$\text{if } (u(n) == 1), BP = BP_P + \eta; \text{ else } BP = BP_P - \eta \quad (3.3)$$

At the end of this clock cycle, 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 η . Indeed, the adaptation step size parameter η is found to affect the quantizer's performance [47]. On one hand, a large η is preferred so as to track rapid fluctuations in consecutive pixel values. On the other hand, a small η is preferred so as to avoid large amplitude oscillations at convergence. To circumvent this problem, we propose to make η 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 η is then multiplied by Λ ($\Lambda > 1$).
- *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 η is reset to its initial value η_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 η is used. Consequently, when $u(n) = u(n - 1)$, η is updated as $\eta = \eta \times \Lambda$ Otherwise, i.e, when $u(n) \neq u(n - 1)$, $\eta = \eta_0$.

Figure 3.4: (a.) Boundary point propagation scheme using Morton (Z) scan [47]. 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 8×8 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.**

3.3.2 Hilbert scanning

The adaptive quantization process explained earlier permits to build a binary image on which quadrant tree decomposition (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 PSNR and compression ratio because **it provides larger spatial correlation, which is favorable for the adaptive quantization and QTD processing.**

Fig. 3.4(a) illustrates a typical Morton (Z) scan which is used to build the corresponding tree as reported in [51]. 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 [51].** Unfortunately, this solution comes at the expense of two additional 8-bit registers for each level of the quadrant. As shown in Fig. 3.4(a), two registers ($A4, B4$) are needed to store the boundary point for the 4×4 quadrant level and two other registers ($A8, B8$) are needed to store those related to the 8×8 quadrant level.

Fig. 3.4(b) illustrates an alternative solution using Hilbert scan sequence. In this scheme, multi-layers hierarchical quadrants are sequentially read-out while maintaining spatial continuity during transitions from quadrant to the next. The storage 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 us-

ing hierarchical address mapping logic. Hilbert scanning is actually composed of multiple levels of four basic scanning patterns as shown in Fig. 3.5.

Figure 3.5: Basic scanning patterns found in Hilbert scan.

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:

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

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

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

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. 3.6. 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. **Secondly, 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 2 Most Significant**

Figure 3.6: 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.

Bits (MSB) 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.

3.3.3 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 [47] 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.** 8-bit register stores data that is going to be written into the on-pixel SRAM, and 4-bit register store the intermediate leaf nodes of a quadrant

Figure 3.7: Block diagram of the shift register at the 4×4 (SR4) and 8×8 (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.

(4 nodes). Therefore, a 12-bit shift register is required. For the sake of clarity, the operating principle of one intermediate level is illustrated as shown in Fig. 3.7. Each valid bit of the shift register SR4 represents the compression information of a 4×4 block. During 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 cannot 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.

In summary, the compression scheme proposed in this work can be generally interpreted as the cascade of 2 basic blocks namely the boundary adaptation block and the QTD processor. The first stage is a lossy compression for which there is a trade-off 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 depending on the input image. The main trade-off involved in this design is related to the first stage in which the number of bits at the output of the adaptive quantizer. A larger number of bits enables improved signal to noise ratio and better quality image 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, 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's not difficult to scale the design procedure.

Figure 3.8: Simulation results illustrating the compression performance (PSNR and BPP) as function of η_0 for Lena image. **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 256×256 and (b) image size = 512×512 .

Operation modes	Size of test images							
	64×64				128×128			
	PSNR	BPP	M	η_0	PSNR	BPP	M	η_0
η_0 -R	21.00	1.01	20.87	26	22.14	0.95	23.43	22
η -R	21.15	1.07	19.79	21	22.42	0.97	23.05	20
η -MZ	21.48	0.97	22.15	18	22.82	0.89	25.64	16
η -SmoothMZ	22.37	0.96	23.37	17	23.83	0.90	26.51	14
η -Hilbert	22.77	0.98	23.11	18	24.26	0.93	26.05	14
η -Hilbert+DPCM	23.06	0.88	26.14	19	24.62	0.81	30.43	15
DCT[52]	19.28	0.91	21.18		30.16	0.80	37.70	
Wavelet[52]	26.78	0.75	35.70		33.01	0.75	44.01	
QTD[47]	31.49	6.56	4.80		30.34	6.13	4.95	

	256×256				512×512			
	PSNR	BPP	M	η_0	PSNR	BPP	M	η_0
η_0 -R	23.66	0.88	26.87	18	25.34	0.84	30.15	15
η -R	24.01	0.93	25.95	16	25.93	0.91	28.52	12
η -MZ	24.32	0.84	28.78	13	25.90	0.85	30.55	11
η -SmoothMZ	25.49	0.86	29.79	12	27.45	0.89	30.76	9
η -Hilbert	26.05	0.88	29.70	12	28.11	0.99	28.27	7
η -Hilbert+DPCM	26.52	0.75	35.25	12	28.53	0.75	38.02	9
DCT[52]	33.98	0.70	48.54		37.08	0.70	52.97	
Wavelet[52]	37.31	0.75	49.75		39.84	0.75	53.12	
QTD[47]	30.10	5.70	5.28		28.88	5.23	5.52	

Table 3.1: Average performance of 20 test images from scientific image database[53] under different operating modes, namely DCT[52], Wavelet Transform[52], QTD[47], fixed η raster scan (η_0 -R), adaptive η raster scan (η -R), adaptive η Morton (Z) scan (η -MZ), adaptive η smooth boundary Morton (Z) scan (η -SmoothMZ), adaptive η Hilbert scan (η -Hilbert) and adaptive η with DPCM using Hilbert scan (η -Hilbert+DPCM). $M = \frac{PSNR}{BPP}$ [dB/BPP]. **For each operating mode, η_0 was optimized in order to achieve the best possible performance.** The η -Hilbert+DPCM mode presents the best PSNR and BPP figures compared to the first generation compression algorithm [47] and for all possible operating modes.

3.3.4 Simulation Results

The performance of our proposed compression scheme i.e adaptive η with DPCM using Hilbert scan (η -Hilbert+DPCM), is compared with other operating modes, namely fixed η raster scan (η_0 -R), adaptive η raster scan (η -R), adaptive η Morton (Z) scan (η -MZ), adaptive η smooth boundary Morton (Z) scan (η -SmoothMZ) [47], adaptive η Hilbert scan (η -Hilbert) for a set of test images from scientific image database [53], as illustrated in Table 3.1. For each sample image different resolutions are generated (64×64 , 128×128 , 256×256 , 512×512), and used in our comparative study. Simulation on different resolutions is important because our implementation is low resolution and the aim of this simulation is to provide some insights on how the processor will perform if we are to increase the resolution. Obviously, the performance of the quantizer is highly dependent on the choice of η [47]. There even exists an optimal η value for a particular image under each operation mode. However, it is **unpractical to tune fine η in order to obtain “optimal” performance (highest PSNR) for each input image.** Therefore, for each operation mode reported in Table 3.1, we sweep the value of η from 5 to 35 and calculate the average PSNR and BPP to find the optimal performance **for the whole data set.** Fig. 3.8 reports the results when sweeping η (5-35) for the “DPCM using Hilbert scan” (η -Hilbert+DPCM) configuration. Both PSNR and BPP are highly dependant upon the value of η . Using a large value for η **enables faster tracking of sharp transients in the signal and hence improved compression ratios are obtained when combined with QTD.** However η cannot be increased indefinitely as it will result in a rapidly degraded PSNR performance. For image sizes of 256×256 and 512×512 , the optimum η was found to be around 13 and 10, respectively. The optimum η is different for different images. An experiment was performed to estimate the value of η based on the statistics of images. The Standard Deviation (STD) of an image was found to be roughly correlated with the η value. The STD values of 55 test images were found and divided by the optimal values of η , which were obtained by the aforementioned process. The results are shown in Fig. 3.9: the Mean value (4.3) and the Root Mean Square (RMS) value (5.1) of the Quotient

Figure 3.9: Standard Deviation vs. Optimal η

(= STD / Optimal η) are plotted on the same figure as well. The Optimal η could be expressed as:

$$\hat{\eta} = \frac{\text{STD}}{Q}, \text{ where } Q \approx 4 \sim 5 \quad (3.4)$$

For different type of scenes, the statistics could vary accordingly, and thus, the range of Q varies. Q values could be trained in different scenes, for practical use, 4 to 5 is the typical range. From Table 3.1, one can notice the benefit of adaptive η by comparing the performance figures in the first and second row, where both modes are based on conventional row and column raster scan. With adaptive η (second row), the PSNR is increased by about 0.5dB. Morton (Z) enables better performance as compared to raster scan **because it is a block based scan which improves 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 [47], 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 clearly shown from this table that using predictive boundary adaptive coding combined with Hilbert scanning (η -Hilbert+DPCM) enables about 25% improvement in terms of performance (expressed by the PSNR to BPP ratio) compared to the first generation design [47]. From Table 3.1, we can also note that significant performance improvements are obtained when using large size images. For our proposed algorithm (η -Hilbert+DPCM), using 512×512 format instead of 64×64 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 3.1 also illustrates a comparison of the proposed algorithm to other standards. One can note that the performance of our processor is clearly superior to a stand-alone QTD and comparable to DCT based compression DCT[52] but clearly inferior to that of wavelet based compression DCT[52]. 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 adaptation processing requiring simple addition, subtraction and comparison for η 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 3.5

3.4 VPIC Image Coding

3.4.1 Visual Pattern Image Coding

VPIC [59] is strongly related to classified vector quantization (CVQ) [60], while there are some major differences. First, the visual patterns are derived directly from psychovisual perceptual models and no data training is required. The same visual patterns are also adopted as edge class in the CVQ in which codebook training has to be performed. Second the pattern matching criteria is not based on the norm error so that tedious norm calculations and searching operations could be eliminated.

In VPIC coding, the visual patterns are represented in binary form, either 0 or 1. The total number of patterns are very few, therefore the memory volume required to store these patterns is small. Each pattern contains an edge information: horizontal, vertical, or $\pm 45^\circ$. These visual patterns are used to code image blocks and represent the original image with high fidelity. The psychovisual redundancy could be exploited when the cortical neurons are sensitive to these visually significant edges while insensitive to other types of edge information.

Assume an image to be coded is represented as an array of pixels $A = P_{m,n}$. The VPIC coding is performed on 4×4 image block $B_{i,j} = [P_{m,n} : 4i \leq m \leq 4i+3, 4j \leq n \leq 4j+3]$. Before assigning a best match pattern to each block, the mean intensity $\mu_{i,j}$ of the image block is coded in the first time. Then the designed visual patterns could be independent of the mean intensity of the block. Unlike the CVQ, where non-edged blocks are classified as *shade*, *midrange*, and *mixed* classes, in VPIC, these non-edged classes are simply treated as uniform pattern. In order to know whether a pattern is a uniform pattern or edge pattern, the variation in terms of pixel intensities has to be measured. There are numerous possible edge patterns in real physical situations, therefore the variation could be difficult to measure. One simple method to measure the variation for edge pattern is the discrete gradient ∇B , which is formulated as:

$$\nabla B = \sqrt{(\Delta_x B)^2 + (\Delta_y B)^2} \quad (3.5)$$

where $\Delta_x B$ and $\Delta_y B$ are approximated directional derivatives and are defined

as:

$$\begin{aligned}\Delta_x B &= \text{mean}(\text{Right half block}) - \text{mean}(\text{Left half block}) \\ \Delta_y B &= \text{mean}(\text{Lower half block}) - \text{mean}(\text{Upper half block})\end{aligned}$$

The gradient orientation is formulated as:

$$\angle \nabla B = \arctan \left(\frac{\Delta_y B}{\Delta_x B} \right) \quad (3.6)$$

The basic visual patterns that are adopted both in VPIC and CVQ literatures are shown in Fig. 3.10. The calculated gradient orientation $\angle \nabla B$ should be quantized into intervals with 45° increments, which consists of -45° , 0° , 45° , and 90° . While for the same pattern, in the same orientation of the edge, the pixel intensities could either increase or decrease. This is defined as the polarity of the pattern and is represented in 1-bit. The negative polarity patterns' orientations are 135° , 180° , 225° , and 270° . Because 1-bit polarity information is enough to invert the existing patterns, there is no need to store the negative polarity patterns. In real applications, not all patterns are always used, usually certain subset of them will be sufficient in use.

There are two boundary points defined: ∇B_{min} and ∇B_{max} . The calculated gradient value ∇B is used to compare with these two thresholds. When $\nabla B < \nabla B_{min}$, the block is termed as uniform pattern. When $\nabla B > \nabla B_{max}$, the gradient ∇B is truncated as ∇B_{max} . When $\nabla B_{min} < \nabla B < \nabla B_{max}$, the gradient is then quantized into a specified number of bits, e.g., 3-bit.

The quantized gradient orientation is used to identify which group of patterns might have a possible best match. The polarity information determines whether the selected patterns should be inverted or not. The pixels that are larger than the mean intensity μ of the block is labelled as 1, otherwise 0. The comparison result for an image block is stored in a binary array. In the basic visual patterns shown in Fig. 3.10, the elements in light blue shades are labelled as 1, otherwise 0. The searching operation is simply binary correlation, which could be realized by XNOR or XOR operations. Take XNOR operation for instance, the binary array is compared with each pattern in a group of patterns with the specific angle and

Figure 3.10: Basic Visual Patterns for 4×4 block: (a) 90° edge patterns, (b) 45° edge patterns, (c) 0° edge patterns, (d) -45° edge patterns

polarity. Each element in the binary array will perform XNOR operation with the corresponding elements in the pattern. The sum value of all the boolean results for each pattern will be obtained and stored. The pattern with the highest sum value will be chosen as the best match pattern.

The block type, the quantized mean intensity value μ , the quantized gradient magnitude ∇B , the pattern index i will be sent to the decoder, where the image is reconstructed. For uniform pattern and edge pattern, the bit allocation is different. Here gives a possible bit allocation scheme for both cases. For uniform pattern, block type 1-bit, mean intensity 6-bit, other parameters could be eliminated. For edge pattern, block type 1-bit, mean intensity 4-bit, quantized gradient magnitude 3-bit, edge polarity 1-bit, and pattern index 3-bit. Therefore, the total number of bits for uniform pattern and edge pattern are 7-bit and 12-bit, respectively.

In the decoding stage, if the block type is uniform pattern, the quantized mean intensity μ could be directly used to reconstruct the block. If it is edge pattern, the reconstructed block is represented by $\mu + \nabla B \times P_i$, where P_i is the pattern with the correct polarity.

3.4.2 Isometry Operations

The isometry operation [61] is to perform a shuffling transformation on an image block. There are eight canonical isometries available, which are simply to rotate or flip pixels in an image block. These isometries are illustrated in Fig. 3.11.

Figure 3.11: Eight Canonical Isometries for 4×4 Image Block

The definition of these isometries are: (a) Identity; (b) Rotate clockwise by 90° ; (c) Rotate clockwise by 180° ; (d) Rotate anti-clockwise by 90° ; (e) Flip about mid-vertical axis; (f) Flip about mid-horizontal axis; (g) Reflect about the first diagonal; (h) Reflect about the second diagonal. These isometry operations could be easily implemented by rotation and reflection.

With isometry operations, a small set of visual patterns could be expanded dynamically. Take the visual patterns shown in Fig. 3.10 into consideration, when the patterns $P_1, P_2, \dots, P_7$ are used, with the help of isometry operations (a), (b), (c), and (d), the patterns $P_8, P_9, \dots, P_{14}$ as well as the negative polarity version of $P_1, P_2, \dots, P_{14}$ could be generated online. The isometry (a) does not include any rotation operation, therefore it is just as simple as fetching a visual pattern from a memory bank, while isometries (b), (c), and (d) involves rotation operations.

The rotation operation could be simply implemented as address mapping, for isometries (b), (c), and (d) the address reordering could be expressed as: $I_b(U_{i,j}) = U_{3-j,i}$, $I_c(U_{i,j}) = U_{3-i,3-j}$, $I_d(U_{i,j}) = U_{j,3-i}$. The isometries (e), (f), (g) and (h) could yield the same patterns as that of (a), (b), (c) and (d). Therefore,

these isometries could be eliminated in the real implementation for these basic patterns. While if more patterns are employed, or when isometry operation is performed on other types of data, the isometries (e), (f), (g) and (h) are still useful.

3.4.3 Simulation Results

The VPIC coding technique involves multiplication and square root operations to calculate gradient magnitude, division and $\arctan$ operation to calculate gradient angle. Therefore it could not be easily implemented for focal plane image compression. The basic patterns $P_1, P_2, \dots, P_7$ could be expanded to different orientations and polarities by the help of isometry operations introduced in Section 3.4.2. Therefore it could facilitate exhaustive search with limited number of patterns stored on the memory, and eliminate the need to calculate the gradient angle as well as to determine the polarity. The pattern search is basically XNOR operation, which is much simpler than performing the division and $\arctan$ operations, especially when the No. of patterns is small.

The gradient magnitude calculation for a block of pixels is hardware and power consuming. When the TFS pixel is adopted for image capture, the pixel values within the block are also naturally sorted in the descending order. Here a modified version of VPIC is proposed for hardware friendly implementation in image compression. An image block is defined as $B_{i,j}$ and its sorted vector is represented as $\vec{P}_{srt}$. The mean value of the block is roughly represented by $\mu = (P_{srt}(8) + P_{srt}(9))/2$. The pixel values larger than the block mean μ are labeled as 1, otherwise 0: $P_{srt}(1, 2, \dots, 8) = 1$, and $P_{srt}(9, 10, \dots, 16) = 0$. The gradient magnitude is roughly represented as $\nabla B = P_{srt}(3) - P_{srt}(14)$. Therefore, there is no need for each pixel to have 8-bit SRAM, while for a whole 4×4 block, 16-bit SRAM storing the block's edge pattern, 32-bit SRAM storing the $P_{srt}(3)$, $P_{srt}(8)$, $P_{srt}(9)$ and $P_{srt}(14)$, and 4-bit Counter are required.

The experimental results for a simplified VPIC scheme for image coding is shown in Table 3.2. Only 2 patterns P_2 and P_6 are used, which only consume 1-bit to represent. Four isometry operations (a), (b), (c), and (d) take 2-bit to represent. The roughly estimated mean μ takes 7-bit to represent. The roughly

Table 3.2: The Experimental Results of the Simplified VPIC for Image Coding in terms of Peak Signal-to-Noise Ratio (PSNR). PP stands for Post-Processing using Median Filter, bpp stands for bit-per-pixel

Images	1 bpp		0.9375 bpp		0.875 bpp	
		PP		PP		PP
Kodie	27.66	28.29	27.65	28.26	27.55	28.15
Zelda	33.29	33.88	33.12	33.70	32.59	33.17
Lena	29.05	29.99	29.00	29.93	28.83	29.72
Peppers	25.26	26.25	25.25	26.23	25.20	26.15
Tiffany	28.26	28.97	28.24	28.94	28.16	28.84
Elaine	30.05	30.70	30.04	30.69	29.82	30.50
Average	<i>28.93</i>	29.68	<i>28.88</i>	29.62	<i>28.69</i>	29.42

Figure 3.12: The edge contour after image reconstruction from VPIC decoding (with no post processing)

estimated gradient magnitude ∇B is represented by 6-bit, 5-bit, and 4-bit, which results in 1 bpp, 0.9375 bpp, and 0.875 bpp, respectively, for a 4×4 block. For each bit-rate, there are 2 columns, the left one is the PSNR result of the decoded image with the VPIC decoding procedure, and the right column **PP** is the result obtained after post-processing using median filter. The average PSNR for 0.875 bpp is only 0.24 dB and 0.26 dB lower than those of 1 bpp before and after the post processing, respectively. For the case of 0.9375 bpp, the average PSNR is 0.05 dB and 0.06 dB lower than that of 1 bpp before and after the post processing, respectively. Therefore the sacrifice in PSNR from 1 bpp to 0.9375 bpp is not as severe as that from 0.9375 bpp to 0.875 bpp. The performance of this algorithm will deteriorate for small images. For example, the PSNR value for *Lena* image

with the size of 256×256 is 3 dB less than that of 512×512 . The simulation program was developed in MATLAB without using any optimization toolbox.

The edge contour after VPIC decoding for *Lena* image is shown in Fig. 3.12. The edges are not finely reconstructed, this is due to the limited patterns used in the compression system. For WSN imaging applications, 0.875 bpp is fairly enough when high quality image is not the utmost desirable feature. The simplified VPIC coding scheme involves very few computations and therefore is hardware friendly. Thus, it could be suited in image compression on the focal plane.

3.5 VLSI implementation

Figure 3.13: (a) Corresponding microphotograph of the chip implemented in Alcatel $0.35\mu\text{m}$ CMOS technology with the main building blocks highlighted. (b) Layout of the pixel.

The single chip image sensor and compression processor is implemented using

0.35 μm Alcatel CMOS digital process (1-poly 5 metal layers). **Fig. 3.1 illustrates the architecture of the overall imager including the sensor and the processor. Fig. 3.13(a) illustrates the corresponding microphotograph of the chip with a total silicon area of $3.2 \times 3.0 mm^2$.** The 64×64 pixel array was implemented using a full-custom approach. The main building blocks of the chip are highlighted in Fig. 3.13(a). The photosensitive elements are n^+p photodiodes chosen for their high quantum efficiency. Except for the photodiode, the entire in-pixel circuitry (Fig. 3.2(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. 3.13(b) 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 [47] ($1.8 mm^2$). 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 3.3, compares the number of flip-flops used in this processor compared to that reported in [47].

Functional Block	This work	[47]
Adaptive η	9	9
DPCM	24	N/A
Smooth MZ	NA	64
QTD	202	1407
Hilbert Scan	0	NA
Total	235	1480

Table 3.3: No. of flip-flops used in this work and [47]. N/A stands for Not Applicable.

3.6 Experimental results and Comparison

In order to characterize the prototype, an FPGA based testing platform has been deployed shown in Fig. 3.14. **The test chip was mounted on a printed circuit board outfitted with an FPGA platform and a 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 firstly configured followed by a global pixel reset signal, which starts the integration process. The timing unit de-counts 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 when all of the data is received. 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. **The chip consumes about 17mW power, in which about 15mW is consumed by the sensor array and 2mW is consumed by the image processor. Table 3.4 summarizes the performance of the chip.**

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

Table 3.4: Summary of the chip performance.

The chip was tested in both compressing and noncompressing modes as shown in Fig. 3.15 illustrating some sample 64 $\times$ 64 images. For the compressing modes, the data from the CMOS image sensor are acquired using the FPGA platform and transferred to the PC for display.

Figure 3.14: FPGA based test platform which is composed of a host computer, FPGA board (Memec Virtex-4 MB) and the chip under test.

Once the data is received, the total number of bits per frame (B_F) is counted and the compression ratio is expressed as: $64 \times 64 \times 8 / B_F$. Fig. 3.15 illustrates some 8-bit sample images as well as compressed sample images with their corresponding BPP value.

Compression scheme	DCT [32]	QTD [54]	Haar Wavelet [34]	Predictive [33]	SPIHT [55]	AQ/QTD [47]	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	$0.5\mu\text{m}$	$0.35\mu\text{m}$	$0.35\mu\text{m}$	$0.35\mu\text{m}$	$0.5\mu\text{m}$	$0.35\mu\text{m}$	$0.35\mu\text{m}$
Array Size	104×128	32×32	128×128	80×44	33×25	64×64	64×64
Processor Area	1.5mm^2*	0.7mm^2*	2.0mm^2*	4.89mm^2	0.36mm^2*	1.8mm^2	0.55mm^2
Power	$80\mu\text{W}/\text{frame}$	$70\text{mW}/\text{chip}$	$26.2\text{mW}/\text{chip}$ $24.4\text{mW}/\text{proc.}$	$150\text{mW}/\text{chip}$ $3\text{mW}/\text{proc.}$	$0.25\text{mW}/\text{chip}$	$20\text{mW}/\text{chip}$ $6.3\text{mW}/\text{proc.}$	$17\text{mW}/\text{chip}$ $2\text{mW}/\text{proc.}$
post-proc requirement	Yes	No	No	No	Yes	No	No

Table 3.5: 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, etc. Estimated areas are marked in Asterisk (*).

Table 3.5 compares the performance of our proposed scheme presented in this thesis with our first generation processor [47] as well as other imagers with compression processors reported in the literature [32], [33], [34], [54], [55]. One should note that the comparison of different compression processors is not obvious as sometimes 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

Figure 3.15: 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.

significantly. In addition, some designs implement only certain building blocks of compression algorithms 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 non-conclusive. 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. This implemen-

tation has the 2nd smallest processor area among all compression schemes in Table 3.5. Among those who do not need external post-processors, this implementation is the smallest in terms of area. The processor consumes the 3rd lowest power among all these reported works. For processors that require no post-processing, it achieves the lowest power consumption. Therefore, the compression processor of this implementation achieves the complete compression task while offering low power consumption and small silicon area.

3.7 Summary

A single chip CMOS image sensor with on-chip image compression processor, based on a hybrid predictive boundary adaptation processing and QTD encoder is reported. 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 our first generation design.** Our compression performance is clearly superior to that of a stand-alone 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 η 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 thesis by introducing a QTD algorithm with pixel storage reuse technique. 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 stand-alone compression processor including predictive boundary adaptation and QTD. A prototype chip including 64x64 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 compres-

sion 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 17mW of power.

This integrated system with a CMOS image sensor and a compression processor falls in the category of sample and then compress paradigm. In the next chapter, a new sensing paradigm that simultaneously senses and compresses image data will be introduced.

Chapter 4

Compressive Sampling

The fundamental limit of conventional image compression algorithms is that all pixel values have to be sampled at or above Nyquist rate in order to avoid signal aliasing. The resulting pressure on Analog-to-Digital Converter (ADC) design has been escalating as image resolution is being rapidly increased and readout speed kept constant. Compressive Sampling, also known as Compressed Sensing (CS) is a newly introduced sampling paradigm that can relax the requirements on the ADC stage. This is because it can perform data compression in the acquisition stage at a sampling rate that is much lower than the prescribed Nyquist rate [63].

In Compressive Sampling, an important requirement for successful signal acquisition is for the captured signal to be sparse. Natural images usually contain redundant information. After transformation, i.e., the Discrete Cosine Transform (DCT) [64] or Digital Wavelet Transform (DWT) [65], its sparseness could be measured. After DCT or DWT transformations, only a few of the complete set of transformed coefficients are significant. The remaining coefficients are either zero or very close to zero. Image compression standards make use of this characteristic to achieve data compression [66]. In these standards, both the significant coefficients and their corresponding addresses are coded for data reconstruction while insignificant coefficients are discarded.

In compressive sampling, the acquired signal does not have to go through such transformations. Compressive sensing assumes the signal to be sparse in a certain transform basis. The number of measurements that have to be acquired for signal reconstruction is directly related to the number of significant coefficients.

The compressive sampling process consists in projecting the signal onto a set of measurements having much fewer elements than the original signal. This process is basically performing an inner product between a set of coefficients and the vectorized pixel values, enabling sampling and compression at the same time.

This non-adaptive linear projection should be as random as possible in order to obtain exact signal recovery. The Gaussian [66] [67] as well as Noiselets [68] [69] random variables can constitute very efficient sensing matrix to project the signal of interest onto desired measurements.

The emerging CS theory is currently receiving a lot of attention in academia. It has been successfully applied to the Magnetic resonance imaging (MRI) system [70] to largely reduce the number of samples that need to be acquired. This drastically reduces the operating costs of MRI system. In the following section, we will introduce the basic theory of CS.

4.1 Compressive Sampling

The sensing mechanism of compressive sampling is to linearly project the original signal $\vec{X}$ to a series of measurements $\vec{Y}$:

$$\vec{Y}_{m \times 1} = \vec{\Phi}_{m \times W} \vec{X}_{W \times 1} \quad (4.1)$$

Where $\vec{\Phi} = [\vec{\phi}_1, \vec{\phi}_2, \dots, \vec{\phi}_m]^T$ is the sensing matrix and each $\vec{\phi}_j, j = 1 \dots m$ is an $W \times 1$ vector. The original signal $\vec{X}$ is vectorized to an $N \times 1$ vector therefore the resulting measurement $\vec{Y}$ is a $m \times 1$ vector and $m \ll W$.

Because the number of measurements K is much smaller than the number of elements N in the original signal $\vec{X}$, the measurement process reduces the dimensionality of the acquired signal. As $m \ll W$, the inverse problem of $\vec{Y} = \vec{\Phi} \vec{X}$ is generally ill posed. The possible number of candidate solutions $\vec{X}'$ that satisfy $\vec{Y} = \vec{\Phi} \vec{X}'$ are not unique by solving the inverse equation:

$$\tilde{X} = \{\vec{\Phi}^T \vec{\Phi}\}^{-1} \vec{\Phi}^T \vec{Y} \quad (4.2)$$

One of the possible solutions that best approximates the original signal is

extracted by the Compressive Sampling framework at a high probability when this signal is sparse enough in a certain sparsifying transform basis $\vec{\Psi}$.

The sparsity of a signal could be exploited by representing it with a vector of coefficients $\vec{\alpha}$: $\vec{X} = \vec{\Psi}\vec{\alpha} \iff \vec{\alpha} = \vec{\Psi}^T\vec{X}$, where $\vec{\Psi} = [\vec{\Psi}_1, \vec{\Psi}_2, \dots, \vec{\Psi}_W]$, and each $\vec{\Psi}_j$, $j = 1, \dots, W$ is an $W \times 1$ vector. If there are only s nonzero coefficients in the vector $\vec{\alpha}$, the original signal could be termed as s -sparse. Here the coefficients with very small magnitude are considered as insignificant or zero coefficients.

Compressive Sampling theory states that when the signal $\vec{X}$ is sparse, it could be approximately/exactly recovered from the measurements $\vec{Y}$ by solving a standard convex optimization problem:

$$\tilde{\alpha} = \arg \min \left\| \vec{\alpha}' \right\|_1 \text{ subject to } \vec{\Phi}\vec{\Psi}\vec{\alpha}' = \vec{Y} \quad (4.3)$$

This l_1 -norm minimization problem could be solved by simple linear programming (LP) technique and there are numerous algorithms and solvers [71, 72, 73, 74] that could be applied to tackle this problem. Usually in these solvers, this problem is equivalent to:

$$\tilde{\alpha} = \arg \min \frac{1}{2} \left\| \vec{Y} - \vec{\Phi}\vec{\Psi}\vec{\alpha}' \right\|_2^2 + \tau \left\| \vec{\alpha}' \right\|_1 \text{ and } \vec{X} = \vec{\Psi}\vec{\alpha}' \quad (4.4)$$

In order to recover the original signal $\vec{X}$ that is s -sparse with a very high probability, the number of measurements m should satisfy the following constraint [75]:

$$m \geq C \cdot \mu^2(\vec{\Phi}, \vec{\Psi}) \cdot s \cdot \log W \quad (4.5)$$

where C is a positive constant and $\mu(\vec{\Phi}, \vec{\Psi})$ is the coherence measure between the sensing matrix $\vec{\Phi}$ and sparsifying matrix $\vec{\Psi}$. The smaller the coherence, the smaller the number of measurements is required. This coherence measure is expressed as:

$$\mu(\vec{\Phi}, \vec{\Psi}) = \sqrt{W} \cdot \max_{i,j} |\langle \vec{\phi}_i, \vec{\psi}_j \rangle| \quad (4.6)$$

This measure is to find the largest correlation between the two matrices. Usually it is in the interval of $\mu(\vec{\Phi}, \vec{\Psi}) \in [1, \sqrt{W}]$. Finding the sensing matrix $\vec{\Phi}$ that has low coherence with the sparsifying matrix $\vec{\Psi}$ and is easy to implement is very

important for hardware implementation of CS.

The random sampling matrix $\vec{\Phi}$ should be highly reusable for a variety of sparsifying matrices with very low coherence. Each measurement carries equally important information for signal recovery. Therefore a signal can be progressively reconstructed with an increasingly higher quality as more and more measurements are acquired. In addition, the nonzero coefficients' values and their corresponding addresses are not required for signal reconstruction. This is in contrast to the transform-based signal compression.

There is another very important property of the sensing matrix $\vec{\Phi}$ for the proof of the above mentioned recovery algorithm. It is referred to as Restricted Isometry Property (RIP) [76]: *there exist an isometry constant $0 < \delta_k < 1$ of a matrix $\vec{\Phi}$ that is related with all possible s -sparse signal $\vec{X}$ satisfying $(1 - \delta_s)\|\vec{X}\|_2^2 \leq \|\vec{\Phi}\vec{X}\|_2^2 \leq (1 + \delta_s)\|\vec{X}\|_2^2$.*

If there are some noises added to the measurements, the original signal could also be recovered by the l_1 -norm minimization with relaxed constraint:

$$\tilde{\alpha} = \arg \min \left\| \vec{\alpha}' \right\|_1 \text{ subject to } \left\| \vec{Y} - \vec{\Phi}\vec{\Psi}\vec{\alpha}' \right\|_2 \leq \epsilon \quad (4.7)$$

The noise source could be white noise, quantization noise [77], or saturation noise [78]. These kinds of noise are, most of the time, found in hardware. This robust signal recovery program could be very efficient in denoising noise contributions introduced in the sensing stage.

The following is an example illustrating how the CS encoding and decoding works. We consider a series of data consisting of 500 elements, of which only 25 elements are nonzero. A 250×500 Gaussian sensing matrix Φ is applied for compressive sampling, generating 250 measurements. Using these 250 measurements, l_1 minimization is performed to exactly reconstruct the original data.

For a 2- D image, the Total Variation (TV) is considered as sparse, and thus the TV minimization program is, in practice, more effective in image reconstruction. The TV of an 2- D image X_{2D} is defined here as:

$$TV(X_{2D}) = \sum_{i,j} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2} \quad (4.8)$$

Figure 4.1: Experiment in sparse signal Compressive Sampling

where $x_{i,j}$ is its pixel in the coordinate (i, j) . Total Variation (TV) norm minimization is equivalent to the l_1 norm minimization of the image gradient. The TV-norm minimization is expressed as follows with $\vec{X}$ being the vectorized version of X_{2D} .

$$\min TV(X_{2D}) \text{ subject to } \Phi \vec{X} = \vec{Y} \quad (4.9)$$

The following is an example illustrating how CS is used to reconstruct a 2D image. The 256×256 image is sparse in the Fourier transform basis Ψ and 6136 (9.36%) Fourier coefficients were sampled. Based on these measurements, the 2D image was reconstructed with l_1 minimization [79] with a Signal-to-Noise ratio of 57.5 dB.

Figure 4.2: Example of Compressive Sampling for a 2D image [63]

4.2 Review of CS Imaging Systems

Researchers from different communities such as signal processing, coding theory, and optics are exploring the development of practical CS optical systems, which would need to be compact and robust. There are three main CS optical architectures: sequential, parallel, and photon-sharing. In sequential architecture, each measurement is taken one at a time. In parallel architecture, all measurements are taken at the same time using a fixed mask. In photon-sharing architecture, micromirror and beam-splitters are used to redirect lights and photons. In the following, we present CS optical systems reported in the literature.

4.2.1 Rice Single-Pixel Camera

The most remarkable work in CS community is the single-pixel camera, in which a single photon detector element is used to capture a scene. The Compressive Sampling stage is performed using a digital micromirror array. The mirror represents a random binary array and reflects the scene to the photon detector. The photons falling onto the photon detector are the aggregation of light intensities falling on individual micromirrors with the right angle. Each micromirror's angle can be controlled to form a different sensing vector. After a series of random projections, the measurement values obtained from the photon detector could be used to reconstruct the original image by the CS reconstruction technique. The key advantage of this system is that it could be readily implemented using any binary sensing matrix, so that the measurements could be directly used to reconstruct the original image with existing CS solvers. Its major disadvantage is that the photon collector, the optics, and even the scene need to be kept still long enough for measurements to be acquired. Although it is possible to rapidly acquire measurements sequentially with shorter exposure time, this degrades the signal-to-noise ratio and would not be suitable for video applications.

The experimental setup for the single pixel camera [68] is shown in Fig. 4.3. Its core element, the digital micromirror device (DMD) array is shown in Fig. 4.4. Examples of reconstructed images are shown in Fig. 4.5.

Figure 4.3: Experimental Setup of the Single-Pixel CS Camera [68]

Figure 4.4: (a) Mirror schematic for digital micromirror device (DMD) from Texas Instruments . (b) Part of the actual DMD array. (Image provided by DLP Products, Texas Instruments.) [68]

Figure 4.5: Single-pixel camera images. (a) 256×256 conventional image of a black-and-white R. (b) Single-pixel camera reconstructed image from $M = 1,300$ random measurements ($50 \times$ sub-Nyquist). (c) 256×256 pixel color reconstruction of a printout of the Mandrill test image captured in a low-light setting using a single photomultiplier tube sensor, RGB color filters, and $M = 6,500$ random measurements. [68]

4.2.2 Application-specific Imagers

Modern spectral imagers typically suffer in performance due to necessary trade-offs between spatial and spectral resolution, especially in photon limited environment. Novel Spectral imagers operating in real-time were proposed in [80, 81]. These imagers were firstly designed using two dispersive elements with binary masks. One dispersive element was subsequently omitted in the second generation design. In the pertaining work [82], compressive spectral image capture was facilitated by illuminating scenes using spatial light modulated tunable spectra light sources. In this latest work on compressive spectral imagers, the light coming from a point in the same line of sight is measured by each pixel [83]. This is termed Compressive Structured Light Code (CSLC). [84] proposes a CS based infrared camera, which drastically reduces optics thickness. A fluorescent CS imaging system for capturing natural sparse images was also developed in [85], to completely eliminate optical lens. CS radar imaging [86], data recovery with translucent media [83], ground penetrating radar[87] and astronomical imaging [88] are other new applications of CS theory. Biomedical applications of CS theory include DNA microarrays [89, 90], MRI [91], and confocal microarray [92]. The analog-to-information converter reported in [93] is developed to directly extract information from the physical layer. The distributed sensing paradigm reported in [94] enables to monitor a number of parameters including temperature,

humidity, or even gas concentration in sensor network applications.

4.3 Practical Challenges to CMOS CS Imagers

The CMOS implementation offers the benefit that the optics used for spatial light modulation could be replaced by circuitry. The reduction in imager size justifies this trade-off. Low cost is another attractive feature for CMOS implementation compared with other approaches that require optical lens which is usually rather expensive. However, there are some challenges like excessive dynamic range, non-negativity in the practical sampling systems, photon noise and developing fast and robust reconstruction algorithms have to be addressed before it could become a mature sensing paradigm. In the following, we review prior works on CMOS CS imagers.

4.3.1 CMOS CS Imagers

Hardware implementation of Compressive Sampling is still in its early stage. Works on compressive sampling for CMOS image sensors have been so far focusing on fast matrix multiplication [95, 96] and random convolution [97, 98] using analog implementations.

Works implementing CS using matrix multiplication have been carried out in the analog domain, to perform separable 2-D transform using noiselet basis $\{1, -1\}$ [68, 69]. The sensing matrix was stored in floating-gate transistors, making the coefficients reprogrammable. Individual pixel currents were multiplied by the stored coefficients using analog vector-matrix multiplier (VMM) [96]. The resulting current is then converted into voltage before being digitized. A small portion of the transformed results are then selected as the compressive sampling measurements. The 2-D transform process is illustrated in Figure 4.6(a) and the computational pixel is also shown in Figure 4.6(b). A matrix was stored in analog floating-gate transistors which form a random access analog memory. Each column of transformation coefficients was selected and buffered out as the input to a block of differential mode computational pixels. The latter convert input differential voltage into differential currents with the gain controlled by the small

Figure 4.6: (a) 2-D Separable Transform; (b) Computational Pixel Sensor [96]

photocurrent induced in each photodiode (Figure 4.6(b)). This row of output differential currents are then input to the fully differential VMM. By performing the separable transform $Y_\sigma = A^T P_\sigma B$, convolution is achieved. The advantage of this approach lies in that it could implement Gaussian or other matrices that require floating point computations. Its limitation is that programming the sensing matrix into floating-gates requires high voltage and specialized equipment. Besides, the small photocurrent makes the databus parasitic capacitance discharge take longer. When performing 2-D separable transform for each block, given each photodiodes needs to integrate once, making measurements for all blocks of an image will take prohibitively longer times.

Other works have implemented CS, in the analog domain, using random convolution, with a noiselet basis as well. A CMOS compressed imager architecture that implemented focal plane random convolution is illustrated in Figure 4.7. The binary coefficients are stored in a looped chain of shift registers. Each shift register is connected to a Passive Pixel Sensor (PPS) to determine the direction of its current. The noiselet coefficients are initialized to a pseudorandom sequence generated by a linear feedback shift register (LFSR). After initialization, these coefficients are shifted along a closed loop. An Op-Amp is used to measure the sum of currents of each column of pixels. A pseudorandom trigger is implemented before the Op-Amp to control when a measurement is to be taken. A multiplexer sequentially feeds the sum values in each column to an ADC for quantization.

Figure 4.7: Random Convolution CS Imaging in CMOS [97]

The quantized values are then accumulated to form the final measurement value. The advantage of this scheme is that it implements CS within the pixel sensor array rather than off-array processing. However, the disadvantage is that taking each measurement requires one integration time period, resulting in prohibitively longer acquisition time for the measurements. The reported time for taking $256^2/3$ measurements was 400ms, which is equivalent to 2.5 frames/s.

A subsequent work implemented random 2-D binary coefficient scrambling [98]. In each pixel, a voltage to current amplifier with regenerative resistor and shift registers for storing the binary coefficient was designed (Figure 4.8). Each shift register is connected with its neighbors and thus random horizontal and vertical shifts, could be achieved. The binary coefficient drives the current from the pixel to the summation column or subtraction column. All these columns of the whole sensor array are then connected to an amplifier and ADC for measurement readout. The advantage of this design is that only one medium bandwidth ADC is required, which is enough to collect all measurements. The disadvantage is

Figure 4.8: 2D Random Scrambling CS Imaging in CMOS [98]

that the pixel has to be reset once before each measurement. Because a period of integration has to be performed after each reset, the actual speed in acquiring all the measurements is limited by the integration time rather than the ADC speed. The integration time for each pixel is determined by the threshold voltage of the reset transistor M_1 , its process variation as well as the resistor mismatch, which can lead to high fixed pattern noise. The latter cannot be removed using the l_1 -norm minimization.

The aforementioned works on matrix multiplication and random convolution directly implemented compressive sensing using noiselet matrix multiplication as the projection of the original image into a sequence of measurements. The computational cost associated to the linear projection is the utmost important issue to be solved before CS could effectively be used for focal plane image compression.

4.3.2 Dynamic Range

A practical issue in CS hardware implementation is the quantization of measurements. Quantization noise [77] could be minimized by designing higher resolution ADCs but this would come at a cost. When the dynamic range of measurement exceeds that of the ADC, large and small measurements values will be simply truncated and represented by the maximum and minimum boundaries of the AD-

C. This inaccuracy caused by quantizer saturation [78] is called granular noise. In the signal reconstruction stage, saturated measurement values could be simply discarded or formulated as inequality constraints to reduce the effect of granular noise.

4.3.3 Non-negativity and Photon Noise

As described in Section 4.1, random sensing matrices need to satisfy the RIP. With bounded or Gaussian noise, the accuracy of reconstruction values could be guaranteed theoretically. These matrices are zero mean, that is half of the coefficients are negative. This could easily lead to negative measurement values. Practical linear optics cannot be applied to construct such kind of system. The total number of photons falling onto the detector is also constrained to be no more than that passing through the aperture: $\|\Phi\vec{X}\|_1 \leq \|\vec{X}\|_1$. A zero-mean sensing paradigm with mean shift to turn every coefficient of the sensing matrix to be non-negative has been adopted in physical implementations [99, 85]. Reconstruction algorithms can be used to mitigate the effect of this shift and compensate it with the setting of high SNR. The successful reconstruction of many CS solvers introduced in the literature heavily relies on this setting. They have the best performance when $\vec{\Phi}^T\vec{\Phi} \approx I$. Assume $\vec{Y}_p = \vec{\Phi}_p\vec{X}$ is measured, where $\vec{\Phi}_p \triangleq \vec{\Phi} - \mu_\Phi$, and $\mu_\Phi \triangleq (\min_{i,j} \vec{\Phi}_{i,j})\mathbf{1}_{m \times W}$. The sensing matrix Φ is zero-mean and Φ_p is non-negative. Thus, $\vec{Y}_p = \vec{\Phi}\vec{X} + \mu_\Phi\vec{X}$. As $\mu_\Phi\vec{X}$ is a constant vector proportional to the sum of all the pixel intensities, one could estimate $\vec{Z} \triangleq \mu_\Phi\vec{X}$ from the data. As a result, the reconstruction algorithms could be applied to $\tilde{\vec{Y}} \triangleq \vec{Y}_p - \vec{Z} \approx \vec{\Phi}\vec{X}$ [99].

With the photon-limited low SNR setting, preserving the light intensity while applying the non-negative sensing matrix poses significant challenges for a CS optical system. The high resolution image generated by CS from photon-limited measurements were evaluated in [101, 102]. The performance of a reconstruction method that minimizes an objective function, which have a penalty term measuring image sparsity and a negative Poisson log likelihood term, for Poisson data, was analyzed under the framework of CS. Error bound was shown to grow with increasing number of measurements with fixed image values [102]. The intuitive

explanation of this result is that dense positive sensing matrix yields measurement values that are proportional to the mean of the sensed image plus a small fluctuation around the mean. It is critical to accurately measure these fluctuations for CS reconstruction, but in photon-limited setting, noise is proportional to the mean of background values, making it difficult to measure the signal.

4.3.4 Reconstruction Methods

Currently a significant amount of research work focuses on finding a fast algorithm for signal recovery. The linear programming problem for CS reconstruction as formulated by Equation 4.4 can be solved using different approaches. However, many software packages for solving linear programming problems are not suitable for images. The Newton's method is the underlying method for many of these solvers. To reconstruct a 512×512 image, a $512^2 \times 512^2$ linear system has to be solved at each iteration. This requires significant memory and computations from the computer. In order to apply gradient-based method, the reconstruction problem stated in Equation 4.4 should be reformulated as the l_1 term is not differentiable. Computational accelerations, such as implementing l_1 regularization by simple thresholding technique are not exploited in most software packages. These issues are being addressed in reconstruction algorithms.

For example, gradient projection methods [103, 104], have reformulated Equation 4.4 to a constrained optimization problem with an objective function that is differentiable. In each iteration, easily computable gradient descent directions are used before being projected to a constraint set. Fast computation can be achieved by means of simple and quick thresholding operations. Iterative shrinkage/thresholding algorithms [105, 106, 106, 107] cast objective functions to a series of simpler optimization programs that can be readily solved by shrinking or thresholding non-significant coefficients in current estimate of α . The matching pursuit (MP) method [108, 109, 110] iteratively processes residuals between Y and $\vec{\Phi}\vec{\Psi}\tilde{\alpha}$ to greedily choose nonzero elements of $\tilde{\alpha}$, which is initialized as $\tilde{\alpha} = 0$. MP algorithms are best suited for problems with little or no noise settings. But the gradient-based algorithms are more robust with noisier settings, and are usually faster than the MP algorithms as $A = \vec{\Phi}\vec{\Psi}$ is not explicitly formed and is only

used for matrix vector inner product computation. Although these algorithms do not hold the property of fast quadratic convergence, they offer better scalability with respect to the size of the image, which the most critical issue for practical CS imaging system.

Chapter 5

Proposed CS Imaging Systems

Prior implementations of CS on CMOS image sensor focal plane, as described in Section 4.3.1, are inherently slow because each measurement takes one frame capture period. In view of a digital implementation, the measured data obtained by multiplying sensing matrix with original signal would require more bits to be represented than the original signal. This may lead to a serious limitation that even though the number of measurements is small, the required memory storage for holding the measurements could be more than that required for holding the original signal.

In this chapter, hardware considerations in the implementation of Compressed Sensing (CS) are examined and a simplified sensing matrix that alleviates the need for multiplication operation in the CS sampling stage is proposed. The sensing matrix is constructed by randomly selecting a number of rows from an Identity matrix. This is effectively sub-sampling a number of pixels randomly from an image. This step is suboptimal in regard to the quality of reconstructed image. However, it is a cost effective approach because the bits required per sample would be the same as in the case of the original pixel value.

The CS scheme is here further extended to address not only the spatial coding space but also the pixel's bit-resolution space. The proposed scheme makes use of the robust recovery property of CS to achieve super bit-resolution imaging. It is related to quantized CS [111, 112]. The quantization error could usually be ignored when high resolution ADCs are employed, while for low resolution ADCs, extra decoding effort has to be put for optimal possible reconstruction [77, 113].

The extreme case of 1-bit quantization, explored in [114], stores only the sign bit of measurement and requires a new decoding algorithm with more constraints for the Convex Optimization (CO).

An hybrid scheme that compressively samples data in both the spatial and bit domain is proposed and validated experimentally in FPGA.

The remainder of this work is organized as follows. Sections 5.1, 5.2 and 5.3 present the proposed spatial domain, bit domain and the hybrid CS techniques, respectively. The description of each of the proposed technique is supported by an experimental validation achieved through the FPGA platform presented in Section 5.4. Finally, a conclusion for this chapter is provided in Section 5.5.

5.1 Spatial Domain CS System

The latest works on the hardware implementation of CS use the noiselet matrix as the sensing matrix [95] [97]. The elements of the noiselet matrix are either 1 or -1 , each with 50% probability. When matrix multiplication is carried out in the analog domain, it requires a fast and high resolution ADC to convert the measurement into a digital representation. With this noiselet matrix, the multiplications could be eliminated. However the summations could make some measurements exceed the dynamic range of the ADC [78].

In a digital implementation, the summation could make each measurement require more bits of storage than the original pixel value. Therefore, when an image is compressively sampled, the total number of bits for representing all the measurements might be more than that of the original image. As a result, there would be no gain in the total number of bits sent from the encoder to the decoder. In addition, the image quality would deteriorate because of undersampling.

To address this, we propose to perform CS in the digital domain, and use a simplified sensing matrix to reduce memory requirements. The sensing matrix is derived from the Identity matrix. Assume there are originally W pixels and m ($m < W$) samples are to be acquired. First of all, an $W \times W$ Identity matrix is constructed. K rows are then randomly selected from this Identity matrix to form a $m \times W$ sensing matrix. This sensing matrix will only have one '1' in each row. As a result, it does not require any multiplication operation.

The sensing matrix construction process is illustrated below. First of all, an Identity matrix $I|_{W \times W}$ is prepared, with $W = 16$ here. From this Identity Matrix, m number of rows are randomly selected, with $m = 4$ in this example.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \boxed{0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \boxed{0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \boxed{0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \boxed{0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

This random sub-sampling matrix for CS becomes:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The equivalent operation of using this sensing matrix is a random sub-sampling of the original image. The computational complexity of different sensing matrices for digital hardware implementation is shown in Table 5.1. The computational complexity of the Gaussian sensing matrix is the highest among all three schemes. The noiselet implementation avoids the multiplication process because the coefficients are either 1 or -1 , but the summation is still required. Additions and multiplications are therefore completely avoided using the proposed sensing matrix. Each measured sample could be represented digitally with the same number of bits as the original pixel intensity. When only 50% of the measurements are acquired, the total number of bits could be compressed to half of the original amount. While for other sensing matrices, the bit/sample is significantly increased. Multiplications and summations are also the source of increased power consumption for the front-end.

Besides this, the coherence between the sensing matrix for random sub-sampling

and the Fourier series is 1. The coherence between this matrix and the DCT is $\sqrt{2}$. Therefore, this sensing matrix for random sub-sampling could be considered as absolutely incoherent with the Fourier transform basis and largely incoherent with the DCT transform basis. This feature guarantees its effectiveness in the sensing stage of CS.

The elimination of multiplication and summation operations achieved by employing the random sub-sampling matrix enables us to address the arithmetic overflow issue as well as the memory storage problem. Therefore, for either analog or digital circuit implementation, the ADC or the bit rate will not be the bottleneck for CS implementation.

Table 5.1: Comparison of the Computational Complexity Between Different Sensing Matrices for Hardware Implementation in Digital Domain (Assume the Sensing Matrix is $m \times W$ and the Bit Per Pixel (BPP) is B and each coefficient in the Gaussian sensing matrix is Q -bit)

Sensing Matrix	Multiplications	Summations	Bits/Sample
Gaussian [66]	$m \times W$	$m \times (W - 1)$	$Q + B - 1 + \lceil \log_2 W \rceil$
Noiselet [68]	0	$m \times (W - 1)$	$B + \lceil \log_2 W \rceil$
This work	0	0	B

Another important feature of this sensing matrix is that it could help save even more power using standby mode. Take the example of a CMOS image sensor, if only 50% of pixels need to be readout, the other 50% could be turned off. In contrast, for Gaussian and Noiselet sensing matrices, no matter how many measurements are required, all pixels have to be turned on, which is in fact contradictory to the spirit of CS.

The linear program that minimizes the total variation of a 2-D image in the spatial domain is proven to be more effective in practice in image reconstruction for compressively sampled measurements. This program can be expressed as [63]:

$$\min TV(\vec{X}_{2D}) \text{ subject to } \vec{\Phi} \vec{X} = \vec{Y} \quad (5.1)$$

where

$$TV(\vec{X}_{2D}) = \sum_{i,j} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2} \quad (5.2)$$

$\vec{X}_{2D}$ is a 2-D image with pixels labeled as $x_{i,j}$. $\vec{X}$ is the vectorized version of $\vec{X}_{2D}$.

Minimizing the Total Variation (TV) norm is essentially equivalent to minimizing the l_1 norm of the image gradient.

In order to randomly select pixel values as the measurement readout, a Pseudo-random Number Generator (PRNG) needs to be implemented. The PRNG is essentially a Linear Feedback Shift Register (LFSR) with XOR gates. The output of the PRNG is either 1 (Select) or 0 (Deselect). It is used to determine whether a pixel should be sampled or not. Once the seed of the PRNG is known, the following sequence could be generated in both the encoder and the decoder. The complete spatial CS system is illustrated in Fig. 5.1.

Figure 5.1: The Proposed Image Compression System with Random Sub-sampling

There are two main modules in this architecture, the pixel array and the PRNG. There are two phases of operation for the pixel array, integration followed by readout. In the integration phase, each photodiode will start to discharge. The current intensity is related to the amount of photons received. With high light intensity, the current will be large and the time for discharging will be short, and vice versa. The time information will be coded by a global counter and a look-up-table (LUT) to generate the pixel intensity and this value is temporarily available in the databus. When the photodiode node voltage drops to a predefined threshold voltage, the pixel intensity in the databus is stored in the local SRAM. The photodiode at this instant will be reset and connected to the power source.

If the light intensity is very low, the charging current will be small as well. When a predefined time period has elapsed and some pixels have still not fired yet, the value zero will be stored in the SRAM of these pixels and their respective photodiodes will be reset. This completes the integration cycle. In the readout phase, the pixel value will be sequentially readout. The output of the PRNG module determines which pixel is to be readout.

The proposed image compression system was implemented and tested on 12 images. The reconstruction quality was found to be lower than with the Gaussian Orthogonal sensing matrix by an average of 2.6 dB. The solver l_1 -Magic [79] used in this work for linear programming requires the input of the sensing matrix $\vec{\Phi}_{m \times W}$, the measurements $\vec{Y}_{m \times 1}$ and the initial guess $\vec{X}_0|_{W \times 1}$ to calculate the best estimation of the original data. Assume the original image data is vectorized into $\vec{X}_{W \times 1}$, the measurements are obtained through the direct projection: $\vec{Y}_{m \times 1} = \vec{\Phi}_{m \times W} \times \vec{X}_{W \times 1}$. The initial guess is obtained through the inverse operation: $\vec{X}_{W \times 1}^0 = \vec{\Phi}_{W \times m}^T \times \vec{Y}_{m \times 1} = \vec{\Phi}_{W \times m}^T \times \vec{\Phi}_{m \times W} \times \vec{X}_{W \times 1}$. When the Orthogonal Gaussian matrix is applied as the sensing matrix, $\vec{\Phi}_{W \times m}^T \times \vec{\Phi}_{m \times W}$ is an $W \times W$ Symmetric Matrix [63].

Table 5.2: Simulation Results Comparing our Proposed Matrix with the Gaussian one

Images	Gaussian	Random Sub-Sampling		
	50%	50%	25%	12.5%
Lena	33.06	31.08	26.90	23.96
Baboon	23.57	23.43	20.96	19.64
Peppers	36.86	31.58	26.76	23.37
Barb	28.32	27.25	23.86	21.18
Watch	30.51	26.88	23.19	20.71
Zelda	33.62	31.81	27.40	24.12
Picnic	33.22	29.36	25.10	22.20
Peppers2	42.36	38.36	32.70	28.45
Parrots	32.75	29.72	26.50	24.47
Tiffany	32.96	29.01	24.78	21.93
Seaview	42.16	40.32	36.78	33.95
Vegetables	23.90	22.73	19.19	17.00
Average	32.77	30.13	26.18	23.41

When the random Sampling matrix employed in this work is applied as the sensing matrix, $\vec{\Phi}_{W \times m}^T \times \vec{\Phi}_{m \times W} = \vec{I}_m|_{W \times W}$ is not an Identity matrix because it has value '1' only on the diagonal but not all elements on the diagonal are '1'. Recall from the process of constructing the sensing matrix $\vec{\Phi}_{m \times W}$, K rows are randomly selected from the Identity matrix $\vec{I}_{W \times W}$. The $\vec{I}_m|_{W \times W}$ matrix contains

just the same m selected rows in its original position, and all its other elements of it are '0's. Therefore $\vec{X}_0|_{W \times 1} = \vec{\Phi}_{W \times m}^T \times \vec{\Phi}_{m \times W} \times \vec{X}_{W \times 1} = \vec{I}_m|_{W \times W} \times \vec{X}_{W \times 1}$ is a truncated version of the original image data. Thus, the initial guess is not complete when feeding it into the decoding solver. This has effectively reduced the constraints imposed on the linear program and this is the reason why the random sampling matrix yields sub-optimal reconstruction quality in terms of PSNR compared with the optimal Gaussian Orthogonal sensing matrix. The other reason is that with an initial guess not close to the original signal, a solution may be found around local minimum. This will provide a sub-optimal solution. However, the merit of this random sub-sampling matrix is that it avoids the matrix multiplication process, which results in high power and area consumption. Table 5.2 compares the PSNR of the reconstructed images using both Gaussian and Random Sub-sampling sensing matrices. When 50% of samples are acquired, the PSNR obtained by the Gaussian matrix is around 2.6 dB higher than by the Random Sub-sampling. 16 bits are required for each measurement (Assume $Q = 5$, $W = 16$, and $B = 8$) by Gaussian sensing matrix, and 8 bits for Random Sub-sampling. This explains the higher PSNR achieved by the Gaussian sensing matrix, and it illustrates the necessary trade off between compression ratio and image quality.

5.2 Bit Domain CS System

CS can be applied to imaging applications not only in the spatial domain, but also in the quantization (bit-resolution) domain. For CMOS Image Sensors with on-chip ADCs, usually the pixel readout speed and the power consumption depend on the pixel's resolution (number of bits) since this will set the specifications for the Analog-to-Digital Converter (ADC). Obviously, high precision ADCs are more difficult to design and require higher power and are generally slower. For digital pixel sensor (DPS), pixel intensities will be converted into a digital representation and stored locally at the pixel level. If high precision is required, the power and area consumption will be monotonically increased, and the readout speed will be proportionally decreased. In addition, acquiring the gray-level intensities using fewer number of bits, results in a non-linear improvement in terms of integration

time, when using time-domain image sensors. This feature is clearly illustrated in Fig 5.2. One can see that the $\frac{\kappa}{I_d}$ relationship to the integration time, makes it possible to improve (in a non-linear way) the integration time when quantizing the sensor's data using fewer number of bits. This makes the sensor more appropriate for high frame rate imaging. The question now is can we make use of CS in order to increase the bit-resolution of an image while maintaining the same rate of bit per pixel (BPP) or more interestingly maintaining the bit-resolution but lowering the BPP.

Figure 5.2: Conversion time for the Nonuniform Time-Domain Quantizer

The Time-to-First Spike Digital Pixel Sensor [115] illustrated in Fig. 5.3 will be discussed to demonstrate the benefits of reduced bit-resolution for image sensing.

The initial photodiode voltage V_n is reset to be V_{rst} . The comparator compares the voltage level V_n to V_{ref} . When the start integration signal SI is pulsed, the signal AR is pulled down and the photodiode voltage starts to discharge. The time for the photodiode to discharge from the V_{rst} to V_{ref} can be approximately modeled as:

$$T_d = \frac{(V_{rst} - V_{ref}) \times C_d}{I_d} = \frac{\kappa}{I_d} \quad (5.3)$$

where C_d is the total capacitance in the photodiode node and I_d is the photo current. When V_n reaches V_{ref} , a pulse will be generated in AR to reset the pixel and this signal will also trigger the Write Enable of the SRAM and the pixel value will be written into the SRAM. The pixel value is generated by a

global nonuniform time-domain quantizer. The pixel value is digitally coded and this code is updated at each time threshold, which could be considered as the quantization level in the time domain. The time it takes to reach the last quantization threshold is termed as the conversion time and it determines the frame rate of the image sensor. If at this time the pixel still does not fire, its value could be directly assigned as zero. After the conversion time, the next cycle could be started by pulsing the SI signal. The conversion time is mathematically expressed as:

$$\tau(n, \Delta I) = \frac{\kappa}{I_{min} + \frac{\Delta I}{2^n}} \quad (5.4)$$

where n is the number of quantization bits and $\Delta I = I_{max} - I_{min}$. I_{max} and the I_{min} are the maximum and minimum photo current, respectively. Assume that I_{min} is very small and it is much smaller than the minimum step size $\Delta I/2^n$ ($I_{min} \ll \Delta I/2^n$). Therefore, I_{min} can be neglected in Equation 5.4. Usually a gray scale image is encoded in 8-bit. If less bits, say 5 bits, are acquired, the conversion time can be shortened compared with the full bit-resolution (8-bit) acquisition:

$$\eta = \frac{\tau(8, \Delta I)}{\tau(5, \Delta I)} = \frac{\kappa / (I_{min} + \frac{\Delta I}{2^8})}{\kappa / (I_{min} + \frac{\Delta I}{2^5})} \bigg|_{I_{min} \rightarrow 0} \approx \frac{\frac{\kappa}{\Delta I} \times 2^8}{\frac{\kappa}{\Delta I} \times 2^5} = 8 \quad (5.5)$$

Equation 5.5 illustrates that the conversion time required by 8-bit quantization is 8 times the conversion time required by the 5-bit quantization. In Fig 5.2, the conversion time for 1-bit, 2-bit, and 3-bit quantizer is shown. The conversion time (τ) for 3-bit quantizer $\tau(3)$ is 2 times that of the 2-bit quantizer and is 4 times that of the 1-bit quantizer.

The shortened conversion time leads to lower power or alternatively higher frame rate and less quantization bits leads to reduced pixel area.

CO was used as the decoding algorithm to reconstruct the image from the compressively sampled measurements. It accomplishes this task by selecting the solution best estimates the original image. The solution with the minimum TV value was chosen as the best solution. Even with noisy observations [116], the original image could still be recovered. This is referred to as Robust Signal

Figure 5.3: Time-to-First Spike Digital Pixel: (a) TFS Digital Pixel Diagram. (b) Timing Diagram: SI, Vn, and AR stand for Start Integration, Photodiode Voltage, and Asynchronous Reset, respectively

Recovery From Noisy Data [63] and is mathematically expressed as follows:

$$\min TV(\vec{X}_2) \text{ subject to } \left\| \vec{\Phi} \vec{X} - \vec{Y} \right\|_2 \leq \epsilon \quad (5.6)$$

When the noise is Gaussianly distributed with bounded variance, the quadratic constraint imposed on the CO program could successfully denoise the sampled data and recover the original image [74]. This characteristic could be very promising for CO in conventional denoising applications when the sensing matrix $\vec{\Phi}$ is an identity matrix. Super bit-resolution can be achieved by considering the bit domain sub-sampling as noise that we attempt to model. For an 8-bit gray scale image, each pixel intensity is represented in 8-bit (in the range of 0 to 255). If only the M -bit MSBs are captured, there will be information lost in the last L -bit LSBs. Assume $M = 6$ and $L = 2$, the M -bit data is considered as the M -MSBs in the decoder, which performs the CO. The lost information is ideally uniformly distributed in the range of $[0, L^2 - 1]$.

$$\begin{aligned} f &= f(M) + f(L) \\ \Rightarrow f(M) &= f - f(L) \\ &= f + (-f(L)) \\ &= f + \text{Asymmetrical Noise} \end{aligned} \quad (5.7)$$

The original image is f and the captured M -bit data is $f(M)$. The lost information $-f(L)$ is the L -bit LSBs. Note that the $f(M)$ is also 8-bit, but with the L -bit

Table 5.3: Comparison of the Acquired Image with only M -MSBs and Its Reconstruction Using Convex Optimization in terms of PSNR with $M=1$ to 7 for 8-bit Gray Scale Image ($f(M)$: The truncated Signal; $\hat{f}$: Recovered Image with the Total Variation (TV) algorithm)

M -MSBs	1-bit ($M = 1$)		2-bit ($M = 2$)		3-bit ($M = 3$)		4-bit ($M = 4$)		5-bit ($M = 5$)		6-bit ($M = 6$)		7-bit ($M = 7$)	
Images	$f(M)$	$\hat{f}$	$f(M)$	$\hat{f}$	$f(M)$	$\hat{f}$	$f(M)$	$\hat{f}$	$f(M)$	$\hat{f}$	$f(M)$	$\hat{f}$	$f(M)$	$\hat{f}$
Lena	11.33	16.86	16.86	22.98	23.00	29.12	29.19	34.69	35.73	40.94	42.69	47.16	51.15	52.37
Baboon	10.82	16.84	16.74	22.53	23.03	29.01	29.26	34.88	35.70	40.93	42.71	47.17	51.16	51.32
Pepper	11.58	17.68	17.10	23.44	22.64	29.14	29.11	34.81	35.68	40.96	42.69	47.19	51.14	52.27
Barb	11.08	16.85	16.87	23.06	23.21	29.33	29.25	34.90	35.69	40.92	42.69	47.16	51.15	51.77
Watch	9.83	16.48	16.39	22.39	22.56	28.69	29.18	34.81	35.73	40.86	42.69	47.16	51.13	52.28
Zelda	11.40	17.60	16.95	23.17	23.05	29.14	29.27	34.89	35.73	40.95	42.69	47.17	51.15	52.07
Picnic	11.92	16.30	17.18	22.91	23.01	28.71	29.46	34.85	35.80	40.96	42.78	47.22	51.13	52.14
Pepper2	10.75	18.50	17.08	23.46	22.63	29.68	29.67	34.51	36.11	40.80	42.92	47.30	51.02	53.28
Parrots	10.29	18.31	16.90	22.72	23.02	29.21	29.17	34.95	35.71	40.99	42.67	47.17	51.16	52.95
Tiffany	9.54	17.12	16.42	22.93	22.97	29.39	29.29	34.98	35.68	40.96	42.71	47.20	51.15	51.96
Seaview	10.01	16.55	16.91	22.49	23.23	29.16	29.26	34.72	35.74	40.87	42.75	47.18	51.15	52.33
Vegetable	11.11	16.46	17.12	22.55	23.17	28.80	29.33	34.76	35.76	40.93	42.75	47.21	51.16	51.39
Avg. Dif.	6.32 (dB)		6.01 (dB)		6.15 (dB)		5.52 (dB)		5.17 (dB)		4.46 (dB)		1.04 (dB)	

LSBs being set to 0. When $f(M)$ is fed to the decoder for signal reconstruction, $-f(L)$ could be treated as noise that is added to the original image. However, the distribution of this noise is not symmetrical around zero, the CO thus cannot remove the noise and recover the original image. In order to address this problem, a fixed offset $(L^2 - 1)/2$ could be added to every captured data and therefore the noise becomes symmetrical around zero and is uniformly distributed in the range of $[-(L^2 - 1)/2, (L^2 - 1)/2]$.

$$\begin{aligned}
f(M + O) &= f - f(L) + \frac{L^2 - 1}{2} \\
&= f + \left(\frac{L^2 - 1}{2} - f(L) \right) \\
&= f + \text{Symmetrical Noise}
\end{aligned} \tag{5.8}$$

where $f(M + O)$ is the M -bit MSB data with the fixed offset and will be fed to the decoder for noise removal.

This signal conditioning process is illustrated in Fig. 5.4. Though after conditioning, this noise is only symmetric around zero but not Gaussianly distributed, the image could still be recovered using CO in practice. Table 5.3 shows the results for images captured with different M -MSBs. $f(M)$ lists the PSNR of the captured image with only M -MSBs when compared with the original 8-bit gray scale image. $\hat{f}$ lists the PSNR of the reconstructed image based on the M -MSBs and signal conditioning using Total Variation program in CO compared with the

original image. From the last row of Avg. Dif. (Average Difference) value, it is clear that there is around 5dB improvement of PSNR when M is about 4 to 6. In this region, the achieved PSNRs are quite high and the conversion time τ is much lower than the 8-bit quantization. Therefore it could be suitable for high frame rate and reduced storage applications. When M is 7, the PSNR is very high for both the $f(M)$ and $\hat{f}$ cases. However, though both the conversion time could be reduced by 2 times, the TV reconstruction program could only increase the PSNR by around 1 dB, which is not significant. Therefore, only the cases when M is 4 to 6 were selected for hardware implementation and validation.

Figure 5.4: Signal Conditioning by Noise Offset before Convex Optimization

According to quantization theory [117], each quantization bit contributes to 6 dB PSNR and therefore for 8-bit gray scale image, the 8-bit quantizer could yield $8 \text{ bit} \times 6 \text{ dB/bit} = 48 \text{ dB}$ PSNR in total. For M from 1 to 6, the PSNR increase is around 5 dB to 6 dB. This indicates that after Signal Conditioning and CO, the actual quantization bits of the acquired truncated pixel value was increased by around 1 bit.

Fig. 5.4 illustrates how CS can be used to generate super-resolution image data. First of all, only the M -MSBs are acquired, which indicates that the L -LSBs are truncated. The L -LSBs are the fine details also called high frequency components of the image. The truncated image is added with a fixed offset as if symmetric noise is added to the original untruncated image. Finally, the CO was performed to remove the noise and reconstruct the original image. Here, the lost information was partially recovered and the reconstructed image is closer to the

original image compared to the truncated image. The quality of the reconstructed image depends heavily on the M -MSBs that are captured. The larger the value of M the more the reconstructed image will be closer to the original image. This illustrates the trade-off involved in the design of our CS system: for TFS DPS, the larger the value of M , the longer integration it takes to capture an image, and the better the quality of the acquired image.

5.3 Hybrid System

A Hybrid system was implemented to explore the case where both spatial domain and bit domain CS are combined. This could be applied to certain imaging systems for which high frame-rate, lower storage and lower power requirements are more important than image quality. Assume the original image is B -bit per pixel with a spatial resolution $\sqrt{W} \times \sqrt{W}$. This image could be rearranged into an $W \times 1$ vector with each data in B -bit ($B = 8$ in this work) format. The hybrid system compressively samples this signal in both spatial and bit domain. m ($m < W$) samples are acquired and each sample is with M ($M < B$) bits format. In this case, an image could still be recovered to a certain extent, using CO techniques.

The sensing matrix Φ used in this hybrid system is basically the same as the one proposed in Section 5.1. Therefore, the CS process for this proposed hybrid system is basically to randomly select pixels as samples from the original image. Here, each sample only contains the M -MSBs of the original pixel. Taking the same example as described in Section 5.1, the sensing matrix for the hybrid CS system is as follows:

$$\begin{pmatrix} 0 & \xi & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \xi & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi & 0 \end{pmatrix}$$

where ξ represents the operation of selecting the M -MSBs of the pixel value. This sensing matrix is only used in the FPGA validation of the system. In the real circuit implementation, this bit selection process could just be eliminated by generating and storing only M -bit data in the on-pixel memory.

The decoding algorithm of the CO used to recover the image is the same as the one used for reconstructing signal from noisy observations, as shown in Equation 5.6. The signal conditioning stage, as discussed in the Section 5.2, is also required before the reconstruction stage in the decoder in order to mimic the noise added to the samples. Table 5.4 shows the PSNR result of the hybrid system with different M -MSBs, with $M = 4, 5, 6, 7$ and 50% of the number of samples. It can be seen that the actual PSNR result is not improved significantly with increasing M . The PSNR value is lower than the stand-alone bit domain CS system described in Section 5.2. Because only 50% of the pixels were randomly sampled, the BPP for each case will be halved. However, the visual perception of the recovered image is quite close to the one recovered from the stand-alone bit domain CS system.

Table 5.4: The PSNR of the recovered image with 50% measurements and with M -MSBs

Images	4-bit	5-bit	6-bit	7-bit
BPP	2-bit	2.5-bit	3-bit	3.5-bit
Lena	29.66	30.69	30.95	31.02
Baboon	23.18	23.35	23.39	23.40
Pepper	30.10	31.18	31.48	31.54
Barb	26.68	27.08	27.18	27.21
Watch	26.29	26.71	26.81	26.84
Zelda	30.33	31.37	31.64	31.71
Picnic	28.43	29.10	29.26	29.31
Pepper2	33.37	36.58	37.73	38.03
Parrots	28.67	29.42	29.62	29.67
Tiffany	28.18	28.78	28.94	28.99
Seaview	33.99	37.79	39.48	39.98
Vegetable	22.47	22.63	22.67	22.68
Average	28.45	29.56	29.93	30.03

These results were obtained from MATLAB simulation. The truncation of bits was simply calculated by $\lfloor P/2^L \rfloor \cdot 2^L$, where P is the pixel value and L is the number of bits to be truncated ($L = 8 - M$). When M is less than 4, the image will be severely distorted and could not be reconstructed by the CS solver (l_1 -Magic). The bit truncation makes the pixel values identical in several dark regions, for which the original pixel values are quite small. Therefore, even with a shifted offset, it cannot mimic the scenario that evenly distributed symmetrical noise is added and thus signal conditioning fails to work.

5.4 Hardware Implementation and Experimental Results

5.4.1 FPGA Implementation of Spatial Domain CS System

The complete random sub-sampling scheme was implemented using an FPGA chip (Spartan-3, XC3S400-PQG208, Speed 4) interfaced to a CMOS image sensor (Fig 5.5). Pixel values are randomly compressively sampled and then transmitted to the PC to reconstruct the image.

Figure 5.5: Testing Platform with FPGA and Camera

Fig. 5.6 shows the reconstructed images from $m = 50\%W$, $m = 25\%W$ and $m = 12.5\%W$ measurements compared with the original image. With $m = 50\%W$, the reconstructed image has quite high PSNR and edges are not visibly blurred. With $m = 25\%W$, the reconstructed image has moderately high PSNR with noticable distortions around edges, especially in regions with high contrast. Image quality with fewer acquired samples further worsens. The image reconstruction was implemented in a host computer with the l_1 -norm minimization solver: l_1 -Magic.

5.4.2 FPGA Implementation of Bit Domain CS System

Bit Domain CS was also validated using the FPGA platform. The original image has an 8-bit depth of resolution. 6-bit MSBs, 5-bit MSBs, and 4-bit MSBs images were acquired using the platform. The acquired data went through the signal

Figure 5.6: Sample images illustrating the experimental validation on FPGA platform for $m = 50\%W$, $m = 25\%W$, and $m = 12.5\%W$ measurements

conditioning process before the CO denoising operation. The original image and the reconstructed images are shown in Fig. 5.7.

When $M = 6$, the reconstructed image is very close to the original image in terms of both PSNR and visual perception. When $M = 5$, pixel values in some regions of the captured image are all the same. This deformation could only be partially recovered after reconstruction. In this case, both the PSNR and the visual perception are acceptable for fast frame rate and low power applications, which do not have very stringent requirement on image quality. When $M = 4$, there are large regions with pixels have the same intensity. This cannot be remedied by CO after signal conditioning. This is because there are 4-bit LSBs removed from the original image. Even after the signal conditioning, the 'noise' still has very large magnitude, as large as $(2^4 - 1)/2 = 7.5$. In dark regions, this magnitude could cause very large signal distortion as the information is largely

lost.

The bit domain imaging is in general achieved by the signal conditioning and CO. The gain of bit-resolution could be as large as around $5db/(6db/bit) = 0.83bit$. When $M = 5$, the captured 5-bit image has lost its information in high frequency components. After the CO, the details of the image could be reconstructed and the whole image could be as good as that is represented by 5.8-bit. In order to obtain good reconstruction quality in terms of both PSNR and visual perception, the magnitude of the noise should be kept low, as in the case of $M = 5$ or 6, the maximum magnitude of noise is around $(2^3 - 1)/2 = 3.5$ and $(2^2 - 1)/2 = 1.5$ respectively. These values are quite small for 8-bit gray scale images thus are suitable for implementation. The noise in this framework is not created, but can be considered as added. Therefore, there is no violation with the foundation of information theory, that is, information can only be processed but not created.

5.4.3 FPGA Implementation of Hybrid System

The hybrid system combines both spatial and bit domain CS, obtained results are shown in Fig. 5.8. When $M = 4$, the lost information in the 4-LSBs cannot be easily recovered, this is because after signal conditioning the noise magnitude (± 7.5) is too large. When $M = 5$, the reconstructed image is similar to its bit domain CS counterpart except for the PSNR, which is much lower. In the case of $M = 6$, the recovered image is slightly blurred compared with the original image but the main features are still clear. This again illustrates the power of CO in signal recovery when incomplete data are acquired [116].

CS is suitable to imaging applications that can trade-off image quality for low power operation. The reconstruction process could be carried out in the decoder, which do not have stringent requirements in terms of power consumption and its computation load.

Resource utilization of the FPGA implementation is summarized in Table 5.5, with a clock rate of $50MHz$. One block RAM (RAMB16_S18_S18) is utilized in the Spartan-III FPGA and the total resource consumed is very low ($\leq 4\%$). This is the result of the simplified sensing matrix proposed in this work.

The FPGA board used is developed by the Opal Kelly company with Spartan

Figure 5.7: Acquired images with M -MSBs and corresponding reconstructed images

III chip on it. The communication control between the FPGA board and the host computer through USB cable could be easily accessed with built-in C++ routines and Verilog/VHDL functions. The number of gates (400K) and block-RAMs (16 block-RAMs, each is 1024×16 -bit) for Spartan III are sufficient to perform CS encoding. Thus, there is no need for more advanced or the latest Virtex/Spartan FPGAs. The utilized Slices, Flip-flops, and LUTs only occupy 3% to 4% of the total resources. For each system, a 1024×16 -bit block-RAM is used to temporary store the image data. The low complexity encoder is the main reason

Figure 5.8: Reconstructed images with 50% samples acquired with M -MSBs

Table 5.5: The FPGA Resource Utilization of the Proposed Systems

CS System	Slices	Slice FFs	4-in LUTs	RAMB
Spatial CS(50%)	118(3%)	134(3%)	165(3%)	1(6%)
Spatial CS(25%)	134(4%)	151(3%)	194(3%)	1(6%)
Bit Domain CS	123(3%)	121(3%)	185(3%)	1(6%)
Bit Domain CS(50%)	147(4%)	162(3%)	205(3%)	1(6%)
Total Resources	3584	7168	7168	16

why the utilized resources are so marginal. A Pseudo-random number generator was implemented in the Spatial CS and Hybrid (Bit Domain CS (50%)) system.

5.5 Summary

An image compression system based on a novel CS framework covering both spatial and bit resolutions was presented. A simplified sensing matrix was first proposed to achieve multiplication-free processing enabling hardware friendly CS implementation. Extensive simulation results revealed acceptable image quality using the proposed sensing matrix, while substantially reducing processing complexity. Bit domain imaging system was realized using the robust signal recovery property of CS. The coarsely acquired signal was treated as a noisy signal which was, after signal conditioning, denoised using CO techniques. A hybrid system which compressively samples both the spatial and the bit spaces was pro-

posed. The signal conditioning process was also performed before the decoding stage using CO by the robust signal recovery technique. The overall system was successfully implemented using an FPGA platform and experimental results validated the proposed system. This work illustrates how CS can be used to provide an attractive trade-off between image quality on one side and BPP, complexity and hence power and silicon area on the other side. In addition, image acquisition is dramatically simplified as it only requires random pixel selection and MSB acquisition. The reconstruction process is carried out on the decoder side, where there are no stringent requirements in terms of power consumption and computational complexity. The proposed scheme could find applications in the area of video sensor networks where complexity at the encoder side is much more critical than at the decoder side.

The application of CS in image compression algorithm is exploited in the next chapter, with CS measurements to reduce the dimensionality of codewords in Vector Quantization compression as well as the number of computations.

Chapter 6

Compressively Sampled Vector Quantization

This chapter explores the application of the Compressive Sampling (CS) framework into the block-based image compression algorithm Vector Quantization (VQ) to reduce the number of Euclidean Distance computations during codebook search.

The compressively sensed measurements of a block is here used to search the closest codeword instead of the block itself. This is done to alleviate the computational burden associated to Euclidean Distance computations and thus boost the codebook search speed. The codebook is also compressed with the same sensing matrix used to compress the image block. This leads to reduced memory storage requirement for the corresponding hardware implementation. Because the image block size is usually not large, the dynamic range of measurements is limited to a small region. This has the effect of easing the stringent requirements put on circuit design.

A predictive partial distance search (PPDS) algorithm for fast searching is also proposed and incorporated into the Compressively Sampled VQ (CSVQ) system, to further reduce the number of computations and increase the codebook search speed.

The remainder of this chapter is organized as follows. Section 6.1 introduces Vector Quantization. Section 6.2 presents CS in Vector Quantization with experimental results reported in Section 6.3. Section 6.4 proposes a predictive partial

distance searching algorithm for fast codebook search. Section 6.5 presents an FPGA implementation of the whole system. Section 6.7 concludes this chapter.

6.1 Vector Quantization

Vector quantization (VQ) is a natural extension of scalar quantization scheme for signal representation [118]. It does not quantize a single source of signal but codes a cluster of signals with predefined templates. When it is applied in image coding, a block (tile) of image of $\sqrt{W} \times \sqrt{W}$ will be vectorized before the template matching process. A typical size of a tile is 4×4 . When the size further increases, say 8×8 , the quality of the coded image becomes affected by the block size. The PSNR, BPP, memory and computations for a VQ system with different block size and codebook size are shown in Fig. 6.1. When the block size becomes larger, the BPP will drastically decrease while the PSNR drops slightly. The memory requirement and computations also increase proportionally with block size. Blocky effect appears when using 8×8 block. Although VQ could achieve the fundamental limit of lossy compression when using large codebook and block size, it would usually be impractical for real world image compression systems. Therefore, the block size is usually set to be smaller than 5×5 and the codebook size is set to be smaller than 512. The PSNR versus BPP for VQ system with the above three different block sizes are shown in Fig. 6.1. For low bit rate coding, a large block size has to be chosen, while for high bit rate coding, a small block size should be used. The size of 4×4 offers the best compromise between the PSNR and BPP.

VQ is illustrated in Fig. 6.2. The templates are all stored in a *codebook* and each template is termed as *codeword* [118]. The template matching process is to search the codeword that is the best proximate of the image block. The metrics of mean square error (MSE) and mean absolute error (MAE) are usually adopted as the template matching criteria. The index of the codebook, which has the lowest distortion with the input vector, will be sent to the decoder, where the same codebook is stored. The decoding process consists in retrieving the codeword according to the received index that represents the vector quantized image block. A typical vector quantization (Encoding) and data restoration (Decoding) process

Table 6.1: Performance Comparison of VQ system with different Block Size and Codebook Size

Block Size	Comparison Metrics	16	32	Codebook Size			
		16	32	64	128	256	512
2×2	PSNR	29.71	31.17	32.05	32.70	33.19	33.83
	BPP	1	1.25	1.5	1.75	2	2.25
	Memory (Byte)	64	128	256	512	1024	2048
	+	48	96	192	384	768	1536
	-	64	128	256	512	1024	2048
	$\times$	64	128	256	512	1024	2048
	$\sqrt{}$	16	32	64	128	256	512
4×4	PSNR	26.04	27.54	28.47	29.55	30.40	31.15
	BPP	0.25	0.3125	0.375	0.4375	0.5	0.5625
	Memory (Byte)	256	512	1024	2048	4096	8192
	+	240	480	960	1920	3840	7680
	-	256	512	1024	2048	4096	8192
	$\times$	256	512	1024	2048	4096	8192
	$\sqrt{}$	16	32	64	128	256	512
8×8	PSNR	22.91	23.54	23.79	23.99	24.15	24.28
	BPP	0.0625	0.078125	0.09375	0.109375	0.125	0.140625
	Memory (Byte)	1024	2048	4096	8192	16384	32768
	+	1008	2016	4032	8064	16128	32256
	-	1024	2048	4096	8192	16384	32768
	$\times$	1024	2048	4096	8192	16384	32768
	$\sqrt{}$	16	32	64	128	256	512

is illustrated in Fig. 6.3.

6.2 CS in Vector Quantization

In vector quantization (VQ) framework, a $\sqrt{W} \times \sqrt{W}$ array would be rearranged into a $1 \times W$ vector, and this vector would then be compared with each codeword in a codebook to search for the codeword with the lowest Euclidean Distance. The dimension of a typical codebook is $L \times W$. When the codebook size, either L or W is large, the computational complexity is high and it takes longer to search. In order to address this problem, many algorithms for fast codebook search have been proposed. For example, partial distance search (PDS) [119, 120, 121] and premature exit techniques [122] could find the codeword with drastically reduced Euclidean distance computations. However, when W is large, evaluating the Euclidean norm would be computationally demanding.

To address this issue, we propose to incorporate the CS framework into the VQ system, to reduce the dimensionality of each codeword. This method, referred to as CSVQ would introduce codeword mismatch but the mismatch errors could be mostly recovered through the convex optimization in the decoder.

Figure 6.1: PSNR vs. BPP for VQ with block size of 2×2 , 4×4 , and 8×8

The dimensionality reduction of each codeword could be achieved with CS encoding. The truncated Hadamard matrix is chosen here as the sensing matrix because there is no multiplication involved and the computational overhead could be alleviated. The sensing matrix Φ is constructed by selecting the first m rows from the Hadamard matrix H . Therefore the new codeword $C_m^i = (\Phi \cdot (C^i)^T)^T$, $i = 1, 2, \dots, L$ could be formed. The vector rearranged from a $\sqrt{W} \times \sqrt{W}$ block of pixels should also be compressively sampled by the encoding matrix Φ before codebook searching.

The original vectorized $\sqrt{W} \times \sqrt{W}$ block is defined as $\nu = \{\nu_1, \nu_2, \dots, \nu_W\}$. After transformation, the vector with reduced dimension becomes

$$\mu = (\Phi \cdot \nu^T)^T = \{\mu_1, \mu_2, \dots, \mu_m\}$$

The vector μ is used to find the closest codeword $C_m^i = \{C_m^{i1}, C_m^{i2}, \dots, C_m^{im}\}$ from the codebook C_m . The Euclidean distance as the distortion measure for this

Figure 6.2: Illustration of Vector Quantization: a tile (4×4) of image is scanned out, vectorized, and compared with each codeword in the codebook

Figure 6.3: Typical VQ Encoding and Decoding Process

version is:

$$d(\mu, C_m^i) = \sum_{j=1}^m (\mu_j - C_m^{ij})^2 \quad (6.1)$$

The index obtained by evaluating the minimum of this distance measure would result in index mismatch. Because the first m Hadamard coefficients do not contain all the energy of the original vector as all the remaining coefficients are not zero. The error rate of index mismatch for different dimensions m is shown in Table 6.2, note that when $m \leq 4$, the Correct Matching Probability ($= 1 - \text{Error Rate}$) is quite low, under 58% for medium to large size codebooks (i.e. $L = 256$ or 512). When codebook size increases, the Correct Matching Probability

will decrease, as can be observed from Table 6.2.

Table 6.2: Correct Matching Probability ($= 1 - \text{Error Rate}$) for Different Dimensions m after Compressive Sampling. N is the Codebook Size

N	Dimension m after Compressing Sampling for 4×4 block														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	0.73	0.75	0.77	0.77	0.94	0.94	0.94	0.94	1.00	1.00	1.00	1.00	1.00	1.00	1.00
32	0.47	0.60	0.65	0.66	0.96	0.96	0.96	0.96	0.99	0.99	0.99	1.00	1.00	1.00	1.00
64	0.38	0.55	0.59	0.60	0.90	0.90	0.90	0.90	0.98	0.98	0.98	0.98	1.00	1.00	1.00
128	0.31	0.47	0.52	0.54	0.86	0.86	0.87	0.87	0.95	0.95	0.97	0.97	1.00	1.00	1.00
256	0.22	0.38	0.44	0.48	0.78	0.79	0.80	0.80	0.93	0.93	0.96	0.96	0.99	0.99	1.00
512	0.13	0.30	0.37	0.41	0.74	0.74	0.76	0.77	0.91	0.91	0.94	0.95	0.98	0.99	0.99
Mean	0.37	0.51	0.56	0.58	0.86	0.86	0.87	0.87	0.96	0.96	0.97	0.98	0.99	0.99	1.00

In the CSVQ system, the codebook stored in the encoder is

$$C_m = \{C_m^1; C_m^2; \dots; C_m^L\}$$

The index sent out to the decoder will be directly used to retrieve the codeword from the original codebook $C = \{C^1; C^2; \dots; C^L\}$. This codeword is not the accurate one and is used as the initial guess of the CS decoding system. The data recovered by convex optimization is thus the reconstructed image. PSNR results after CS reconstruction are given in Table 6.3. It could be seen that when $m = 4 \sim 6$, the PSNR remains high enough while computational complexity has been significantly reduced. For $m = 4$, the PSNR sacrifice is 1.7 dB while the computational complexity, in terms of the number of additions and multiplications, is only 1/4 of the normal VQ. When m increases, the PSNR could be improved, while the computational complexity increases as well. When $m \geq 9$, the increase in PSNR is trivial with increasing value of m , and when $m \geq 4$, the PSNR sacrifice compared with the Normal VQ is less than 1.7 dB, which is quite a small difference. Therefore, values in the range $4 \leq m \leq 8$ are preferred.

The encoding and decoding process is reiterated as follows: each W codeword is compressed by an $m \times W$ ($m \ll W$) sensing matrix Φ to get an $m \times 1$ compressed codeword. The whole compressed codebook, in the size of $L \times m$, is stored in the encoder. Each image block in the encoder is firstly compressed by the same $m \times W$ sensing matrix and then compared with every compressed codeword. The MSE (or MAE) is chosen as the distance measure in codebook search. The index of the closest codeword is also sent to the decoder, where the corresponding original full length codeword is retrieved to construct the image, which is labeled as CSVQ in Fig. 6.4. This codeword could also be used as the initial guess for the

Table 6.3: PSNR for CSVQ for a 4×4 block when Codebook Size is 512

Dimension m	Normal VQ (dB)	VQ Reduced Dimension (dB)	CS Recover (dB)
1	29.83	16.72	26.10
2	29.83	18.51	26.99
3	29.83	19.94	27.29
4	29.83	21.39	28.12
5	29.83	24.51	28.11
6	29.83	25.14	28.56
7	29.83	25.50	28.66
8	29.83	25.40	28.72
9	29.83	28.62	29.21
10	29.83	28.63	29.27
11	29.83	29.08	29.45
12	29.83	29.19	29.52
13	29.83	29.79	29.75
14	29.83	29.79	29.75
15	29.83	29.82	29.78

convex optimization data reconstruction if better image quality is desired by the end user. The reconstructed image from convex optimization program is labeled as CSVQ_Rec, as illustrated in Fig. 6.4. The basic principle of this hybrid system

Figure 6.4: Diagram illustrating CS in Vector Quantization for Image Compression: the output CSVQ is the directly constructed image using the original full length Codebook; the output CSVQ_Rec is the reconstructed image from Convex Optimization (The l_1 -norm minimization program)

is simple. At the encoder side, only the transformation of each block has to be added while other operations such as distance measure calculation and memory retrieval from codebook are the same as in the conventional VQ. At the decoder side, the normal data construction is performed. Besides, data recovery by l_1 minimization could also be carried out.

The reason why CS measurements could be used for codebook search lies on the findings in [126], which suggest that CS measurements could be used

for pattern recognition. VQ encoding is essentially a relaxed pattern matching operation using minimum distance search. Now the question is: is it worthwhile to have extra computational overhead to calculate the measurements (matrix-vector multiplication) so as to have less computations in the codebook search? If all the elements $\phi_{i,j}$ in the sensing matrix Φ are floating point numbers, the matrix-vector multiplication would require a dedicated multiplier (i.e. 16-bit) and an accumulator, resulting in large silicon area and power consumption but also reduced data processing speed. On the other hand, if the elements of the sensing matrix Φ are integer (i.e. Integer-DCT), the matrix-vector multiplication could be simplified to shift and addition operations.

6.3 Experimental Results for the CSVQ System

The proposed VQ encoding scheme was evaluated using the standard images *Lena*, *Peppers*, *Kodak*, *Tiffany*, *Zelda* and *Elaine* with 256 gray scales (8-bit). A codebook was generated using Linde-Buzo-Gray (LBG) algorithm with 65 standard images as the training dataset, which does not include the above six testing images [118]. The image block size was $W = 4 \times 4 = 16$, and the original codebook size was $L \times W = 512 \times 16$. The first m -rows of the Hadamard matrix (in the size of $m \times W$, where $m \leq W$) was used as the sensing matrix Φ to compress both the input image block and the codebook. The reason for choosing the Hadamard matrix was motivated by its elements being either $+1$ or -1 . This greatly simplifies the matrix-vector multiplication because it only requires an accumulator. In order to limit the memory size of the codebook, the magnitude of each codeword and measurement was scaled down to be 8-bit, with the LSBs discarded. The compressed $L \times m$ codebook was directly stored in the encoder. In Table 6.4, image quality for different codebook sizes (m) for CSVQ and corresponding reconstructed images CSVQ-Rec by convex optimization are presented. When the m value is small, the image quality enhancement after CS reconstruction is significant. However, when the m value is large the PSNR obtained from CSVQ is already quite high. The computational complexity as well as memory requirement compared with normal VQ are presented in Table 6.4. When MSE is used as the distortion measure for both schemes and $m = 4$ is chosen, the memory

Table 6.4: Image Quality Results in terms of Peak Signal-to-Noise Ratio (PSNR) for *Lena* with Different No. of Measurements: CSVQ is constructed from the original codebook; CSVQ_Rec is reconstructed from l_1 -norm minimization

No. of Measurements: m			1	4	8	12	16(VQ*)
PSNR	Lena	CSVQ	16.95	19.2	24.35	30.69	31.15
		CSVQ_Rec	26.98	29.91	30.45	31.11	31.15
	Peppers	CSVQ	17.15	19.46	26.05	28.78	30.18
		CSVQ_Rec	24.69	26.64	27.79	29.02	30.18
	Average	CSVQ	17.39	21.80	26.74	31.12	31.72
		CSVQ_Rec	28.06	30.01	30.53	31.36	31.72
Computation	$\times$	This work	L	$4L$	$8L$	$12L$	$16L$
		VQ	$16L$				
	$\pm$	This work	$L + 15$	$7L + 15 \times 4$	$15L + 15 \times 8$	$23L + 15 \times 12$	$31L + 15 \times 16$
		VQ	$31L$				
	Compare	This work	L				
		VQ	L				
	Memory	This work	L	$4L$	$8L$	$12L$	$16L$
		VQ	$16L$				

access, the multiplication and additions for this work are around 1/4 of normal VQ. Comparison operations are the same in these two cases. The average PSNR for $m = 4$ is 30.01dB, which is visually acceptable. Therefore, for the CSVQ encoder, $m = 4$ is a viable choice for codebook storage and data encoding. The quality of the *Lena* image with different values of m could be observed in Figure 6.5. Note that the quality of the recovered images improved after the l_1 norm minimization.

6.4 Fast Searching Algorithm

The proposed CSVQ framework could also be applied to the PDS fast searching algorithms in Hadamard domain [123, 124]. These algorithms accelerate the searching speed with the help of some premature exit criteria. The compressed codeword C_m is used here to illustrate the fast searching process. The PDS in the Hadamard domain utilized the properties of the Hadamard transform as well as the triangular inequality. The first property of Hadamard transform is that $\mu_1 = \nu_1 + \nu_2 + \dots + \nu_W$. The first element of μ is the sum of all elements in the vector ν . The second property is related to the triangular inequality:

$$\begin{aligned}
d(\mu, C_m^i) &= \sum_{j=1}^m (\mu_j - C_m^{ij})^2 \\
&= (\mu_1 - C_m^{i1})^2 + \sum_{j=2}^m (\mu_j - C_m^{ij})^2 \\
&\geq (\mu_1 - C_m^{i1})^2
\end{aligned} \tag{6.2}$$

		No. of Measurements				
		$m = 1$	$m = 4$	$m = 8$	$m = 12$	$m = 16$
CSVQ
	CSVQ_Rec

Figure 6.5: Reconstruction CSVQ Results for Lena Image for Different No. of Measurements m

This inequality could be expressed as follows:

$$\begin{aligned} |\mu_1 - C_m^{i1}| &\leq \sqrt{d(\mu, C_m^i)} \\ C_m^{i1} - \sqrt{d(\mu, C_m^i)} &\leq \mu_1 \leq C_m^{i1} + \sqrt{d(\mu, C_m^i)} \end{aligned} \quad (6.3)$$

The preparatory work for fast searching is to sort the codebook C_m according to the value of C_m^{i1} in an ascending order. This operation only needs to be performed once. The operation of this Hadamard domain PDS algorithm is summarized in pseudocode as follows [124]:

1. Initialize a closest codeword C_m^p that satisfies

$$p = \arg \min_i |\mu_1 - C_m^{i1}|, \quad i = 1, 2, \dots, L \quad (6.4)$$

2. Calculate the following parameters:

$$\begin{aligned} d_{min} &= d(\mu, C_m^p) = \sum_{j=1}^m (\mu_j - C_m^{pj})^2 \\ MIN &= \mu_1 - \sqrt{d_{min}} \\ MAX &= \mu_1 + \sqrt{d_{min}} \end{aligned}$$

3. Loop A: for $i = p, p-1, p+1, p-2, p+2, \dots$ (Alternating Up (-) and Down (+) directions)

if ($MIN \leq C_m^{i1} \leq MAX$)

Loop B: for $q = 2, 3, 4, \dots, m$

$$d^q(\mu, C_m^i) = \sum_{j=1}^q (\mu_j - C_m^{ij})^2$$

if ($d^q(\mu, C_m^i) > d_{min}$)

C_m^i rejected; continue;

else

if($q == m$)

Update $d_{min} = d(\mu, C_m^i)$, MIN , MAX

$Idx = i$;

else

if ($C_m^{i1} \leq MIN$)
 Stop in the Up (-) direction
 if ($C_m^{i1} \geq MAX$)
 Stop in the Down (+) direction
 C_m^i rejected; continue;

4. The *Idx* value is used for data reconstruction

In the first step, the computational effort in the searching process described by Equation (6.4) can be significant when the codebook size L is large. To address this, we propose a predictive fast searching PDS algorithm, and we refer to it as PPDS algorithm. The index p is obtained from a block's neighboring blocks. In [121], the criteria of Sum of Squared Error (SSE) is used to determine the closest neighbor. In [125] the absolute potential difference is used.

$$SSE \equiv \sum_{i=1}^k (\nu - \hat{\nu})^2 \quad (6.5)$$

The potential of a vector ν is defined in [125] as:

$$P(\nu) \equiv \sum_{i=1}^k \nu^2 \quad (6.6)$$

The absolute potential difference is thus defined as:

$$|P(\nu) - P(\hat{\nu})| = \left| \sum_{i=1}^k \nu^2 - \sum_{i=1}^k \hat{\nu}^2 \right| \quad (6.7)$$

In both works [121, 125], the prediction schemes are based on the assumption that the codebook is sorted in ascending order of potential values of each codeword. In our algorithm, the codebook is sorted in the ascending order of the first transformed coefficient, that is, the sum of the original codeword. Therefore in our work, the prediction is based on the difference value of the first transformed coefficient and the above two criteria do not apply. The alternating searching direction simplifies to a single direction once the closest codeword is determined. Boundary conditions are also considered. The whole process is detailed below:

1. Initialize a closest codeword C_m^p that satisfies:

B	C
A	X

$$p = \arg \min_i |\mu_1 - C_m^{i1}|, \{i = i_A, i_B, i_C\}$$

X is the current block, i_A , i_B , and i_C are the index of neighboring blocks

2. Determine the direction:

If $((\mu_1 - C_m^{p1}) \leq 0)$

$$dir = -1 \text{ (Going Up)}$$

else

$$dir = 1 \text{ (Going Down)}$$

3. Calculate the following parameters:

$$d_{min} = d(\mu, C_m^p) = \sum_{j=1}^m (\mu_j - C_m^{pj})^2$$

$$MIN = \mu_1 - \sqrt{d_{min}}$$

$$MAX = \mu_1 + \sqrt{d_{min}}$$

$$Idx = p \quad /* \text{ Initialize the index} */$$

4. Loop A: for $it = 1, 2, 3, \dots$

$$i = p + dir \times it$$

if $(i \leq 1 || i \geq L)$ Stop

$$Dtp = 0 \quad /* \text{ Current Distortion set to 0 } */$$

if $(MIN \leq C_m^{i1} \leq MAX)$

Loop B: for $q = 1, 2, 3, \dots, m$

$$Dtp = Dtp + (\mu_q - C_m^{iq})^2$$

if $(Dtp > d_{min})$

$$C_m^i \text{ rejected; continue;}$$

else

if $(q == m)$

$$\text{Update } d_{min} = d(\mu, C_m^i), MIN, MAX$$

$$Idx = i;$$

else

if $(C_m^{i1} \leq MIN)$ Stop

if ($C_m^{i1} \geq MAX$) Stop
 C_m^i rejected; continue;

5. The Idx value is used for data reconstruction

The prediction scheme presented above requires a memory bank to store the previous index values. In order to address this problem, we proposed a new scheme that does not consider the block B . Only blocks A and C will be used to find which one is the closest to the current block X . By doing so, only one row of memory is required to store the previous indices. The process is illustrated in Fig. 6.6. All the contents in the memory are initialized to a fixed number, say $L/2$. When the current block is in the left border of an image, only the block C , which is the first element in the memory, is used as the prediction. After the fast searching process, Idx is stored back to the memory in the original position. When scanning to the right, both the indices of blocks A and C are used for prediction. The index for block A is obtained from the previous searching result. The index C is obtained from the searching result of the previous row. Therefore, A is equivalent to the left neighbor and C is equivalent to the upper neighbor. The Idx is again stored back to the memory. Thus, the content in the memory is continuously updated along with the raster scanning of the image.

Figure 6.6: Implementation of the Proposed Modified Prediction Scheme: i_A and i_C are the stored codeword indices in the previous scan, Idx is the index of the best matched codeword obtained from the proposed PPDS algorithm

Table 6.5 illustrates the effectiveness of the proposed PPDS algorithm in reducing the average number of searches per block and in enhancing the image quality in terms of the PSNR. The number of Searches of the PDS algorithm is higher than the proposed PPDS algorithm. This could be explained by the

alternating searching directions it employs. The proposed PPDS algorithm also outperforms the PDS algorithm in terms of PSNR. This is due to the intrinsic problem induced by the codebook sorting. The codebook is sorted by the first element in each codeword after the Hadamard transform. The first element is the sum of all the elements of the original codeword. This is different from the codebook sorting principle stated in [125], that is, to sort the codebook by its potential or l_2 norm. The potential value contains the information of both the Mean and the variance. However, the sum value only contains information of the Mean. Therefore, even though the codeword with the best matched sum value is found, by searching around it, it is still possible that the optimal codeword cannot be found. Fig. 6.7 shows the potential values of the sorted codebook. It is clear that in some cases, the potential values differ a lot between adjacent codewords. This could also explain the fact that the PSNRs obtained by this fast searching algorithm are slightly less than that obtained by the full search algorithm.

The modified prediction scheme using only A and C costs around 2 more searches than the classic approach using A , B and C . While the PSNR is 0.02dB higher for the modified scheme than the classic scheme. Less neighbors would make the prediction less accurate and therefore more searches are required. The irregularity caused by codebook sorting again plays an important role here. More searches could make the searching result closer to the optimal.

Table 6.5: Comparison between the Original PDS Algorithm [124] and the Proposed PPDS Algorithm with Different Prediction Schemes

Images	Original PDS[124]		Proposed PPDS, Prediction with A, B , and C			
	No. of Searches	PSNR (dB)	No. of Searches	PSNR (dB)	No. of Searches	PSNR (dB)
Zelda	26.9	32.13	25.2	33.51	26.6	33.54
Lena	30.7	29.16	29.2	31.01	30.8	31.05
Barb	62.0	22.54	47.8	23.16	50.4	23.19
Peppers	31.6	28.24	29.6	30.08	31.0	30.11
Elaine	35.9	29.07	30.4	30.81	32.2	30.83
Average	<i>37.4</i>	28.23	<i>32.4</i>	29.72	<i>34.2</i>	29.74

When $m = W$, the proposed PPDS algorithm is identical to a fast searching scheme. For $1 \leq m \leq W$, the searching speed in Loop B could be accelerated. The smaller the value of m , the faster the searching speed. The mismatch induced here could be recovered by a CS decoder. The PSNR and the Average No. of Searches for each block are shown in Table 6.6. When $m \geq 3$, the PSNR obtained

Figure 6.7: Potential Values of the Sorted Codebook: Sorted by the First Elements after Hadamard Transform

by CS recovery is quite satisfactory. When $9 \leq m \leq 12$, the CS recovery does not boost the PSNR significantly. When $m \geq 13$, the PSNR obtained by CS recovery is seen to be slightly smaller than that before the reconstruction.

Table 6.6: PSNR and the Average Number of Searches for FSVQ with 4×4 Block When Codebook Size is 512. The Prediction uses Only A and C for Hardware Friendly Implementation

Dimension m	VQ Reduced Dimension (dB)	CS Recover (dB)	No. of Searches
1	16.62	26.37	15.5
2	18.27	27.39	18.2
3	18.95	28.16	21.6
4	19.17	28.45	24.1
5	23.58	28.52	26.8
6	23.63	28.68	27.1
7	23.99	28.80	27.7
8	24.04	28.88	28.1
9	28.49	29.26	30.1
10	28.69	29.44	30.6
11	29.03	29.49	31.8
12	29.08	29.50	32.6
13	29.73	29.71	33.4
14	29.73	29.71	33.5
15	29.74	29.72	33.9
16 (Full)	29.74	29.74	34.2

Comparison of computational Complexity is summarized in Table 6.7. When

m increases, the No. of additions, subtractions and multiplications increases accordingly. In the original PDS algorithm, Equation 6.4 requires a significant number of subtraction operations. The proposed PPDS algorithm significantly reduces subtraction operations. When $m \leq 4$, the number of additions, multiplications, and sqrt operations for the PPDS algorithm is larger than the PDS algorithm. When $5 \leq m \leq 8$, the No. of these operations for PDS is slightly larger than that of the PPDS. When $m \geq 9$, the No. of these operations for PDS is more than that of the PPDS. In general, the computational complexity of the PPDS algorithm is smaller than the PDS algorithm. The proposed PPDS with predition using only A and C requires slightly more computations than that using A , B , and C .

6.5 Architecture

This proposed CSVQ with PPDS image compression scheme was implemented in FPGA. The proposed architecture is sketched in Fig. 6.8 for illustration. There are mainly two parts: one is a pixel array and the other is the compression engine. The pixel is a Time-to-First-Spike (TFS) Digital Pixel Sensor (DPS) that integrates local memory. Pixel values are read out through column and row decoders.

Pixels within the same 4×4 block are read out serially and then fed into the truncated Hadamard transform processor. The original Hadamard matrix should be 16×16 for 16 pixels. In the truncated Hadamard matrix, the No. of rows, which is equal to the No. of measurements m , is set to 4. Therefore, this Hadamard transform processor can be implemented as illustrated in Fig. 6.9, where the Hadamard coefficients are stored in the registers. The input is noted as Img , which is the readout value from the image sensor array. Its value is updated every clock cycle. The basic building block of this processor is an Accumulator, which consists of an Adder/Subtractor and a Register. As the Hadamard matrix consists of 1 or -1 , matrix multiplication could be realized merely by adding and subtracting. Therefore, after 16 clock cycles, the output values μ_1 , μ_2 , μ_3 , and

Table 6.7: Comparison of the Computational Complexity of the Three Fast Searching Schemes

Dimension m	Original PDS [124]				Proposed PPDS Fast Searching Algorithm, Prediction with A , B , and C							
	Add	Sub	Mult	Sqrt	Add	Sub	Mult	Sqrt	Add	Sub	Mult	Sqrt
1	2.2	515.2	2.2	1.0	26.9	30.8	14.5	13.3	27.9	31.1	15.1	13.8
2	17.0	530.0	15.4	2.6	39.4	43.4	34.3	6.2	40.8	44.0	35.2	6.7
3	43.9	556.9	42.0	2.9	64.4	68.3	61.0	4.3	66.9	70.1	63.4	4.5
4	77.2	590.2	75.2	3.1	94.3	98.3	91.3	4.0	97.1	100.3	93.9	4.2
5	119.8	632.8	117.4	3.3	127.8	131.8	125.7	3.1	131.1	134.3	128.8	3.3
6	147.2	660.2	144.9	3.3	154.9	158.9	152.9	3.1	158.8	162.0	156.4	3.3
7	179.6	692.6	177.2	3.4	184.5	188.4	182.4	3.0	188.9	192.1	186.6	3.3
8	211.1	724.1	208.7	3.4	213.6	217.5	211.6	3.0	218.7	221.8	216.4	3.3
9	271.0	784.0	268.5	3.5	257.0	261.0	255.5	2.6	265.5	268.6	263.7	2.7
10	311.9	824.9	309.5	3.5	291.0	295.0	289.4	2.6	300.3	303.5	298.6	2.7
11	366.8	879.8	364.3	3.5	332.1	336.0	330.6	2.5	342.8	346.0	341.2	2.7
12	420.2	933.2	417.7	3.5	372.4	376.4	370.9	2.5	384.2	387.3	382.5	2.6
13	473.8	986.8	471.4	3.4	412.8	416.7	411.3	2.4	423.8	427.0	422.3	2.6
14	514.6	1027.6	512.1	3.4	446.6	450.6	445.2	2.4	458.4	461.6	456.9	2.6
15	562.8	1075.8	560.4	3.4	484.3	488.2	482.8	2.4	496.5	499.7	494.9	2.6
16 (Full)	607.8	1120.8	605.3	3.4	520.3	524.2	518.9	2.4	533.6	536.7	532.0	2.6

μ_4 are ready for the next stage of computation.

The architecture of the Absolute Difference Accumulator is shown in Fig. 6.10(a). Each element of a specified codeword (codeword) is fed into the node ' C_m ' as one input. The computed Hadamard coefficients μ_1 , μ_2 , μ_3 , and μ_4 are serially fed in as the other input. After 4 clock cycles, the output d , which is the Mean Absolute Error (MSE), is ready for comparison in the search module.

Figure 6.8: Architecture of the proposed image compression scheme that incorporated Compressive Sampling in Vector Quantization

In both the Predictor and the Absolute Difference Accumulator, taking the difference and then the absolute are essential stages. Taking the absolute value is equivalent to taking the two's complement. Fig. 6.10(b) illustrates a digital implementation of 2's complement operation for 16-bit data [127]. This implementation avoids the use of a full adder, which requires 28 transistors (T). In a typical implementation, MUX (6T), NOT (2T) and Full Adder (28T) are required, which costs $16 \times (6 + 2 + 28) = 576T$. In this implementation, four types of gates are required: XOR (6T), AND (6T), NOT (2T), and MUX (6T). The total transistor cost is $16 \times (6 + 2 + 6) + 15 \times 6 = 314T$, which corresponds to 45% saving in transistor count.

Timing control is complex as different modules have to be enabled or disabled following a predefined timing sequence. The complete image compression system was implemented in FPGA, where an extra piece of block-RAM is allocated for temporarily storing pixel values. which emulates the pixel array. The Xilinx Spartan-III XC3S400 device with package PQ208 and speed -4 was adopted for the FPGA implementation. The FPGA board (Opal Kelly) is commercially

Figure 6.9: The architecture of the Truncated Hadamard Transform Processor: $m = 4$ rows of the Hadamard matrix are retained for transformation, Img is the pixel readout value, μ_1 , μ_2 , μ_3 , and μ_4 are the output coefficient values, which are ready after 16 clock cycles.

Figure 6.10: (a). Absolute Difference Accumulator; and (b). Digital implementation of two's complement

available and supports Universal Serial Bus (USB) data transmission protocol, which facilitates the code development. The FPGA resource utilization is small, as shown in Table 6.8.

Table 6.8: FPGA Resource utilization for the CSVQ system

Resources	Used	Available	Utilization
No. of Occupied Slices	436	3584	12%
No. of Slice Flip Flops	437	7168	6%
No. of 4 input LUTs	630	7168	8%
No. of Block RAMs	3	16	18%

The chosen bus width was 16. The reason for making this choice is twofold. First of all, sign extension of the 8-bit pixel value yields a 9-bit signed binary number. Taking the truncated Hadamard transform results in 13-bit data. Secondly, the width of the Block RAM is 16-bit for each entry. Therefore, choosing the width of the data bus to be 16-bit could be more than sufficient for processing the transformed coefficients. The pixel values are fed into the Block RAM through USB transmission from the host PC. The size of each Block RAM is 1024×16 -bit, which means that there are 1024 entries, and each entry is 16-bit.

The test image size was 128×128 . There are thus 32×32 blocks in total, with each block being 4×4 . In order to process the whole image, the pixel values have to be fed into the Block RAM $128^2/1024 = 16$ times.

The codebook is stored in the Block RAM as well. The size of the Block RAM is $1024 \times 16 = 256 \times 4 \times 16$. Therefore, there are 256 codewords. Each codeword contains 4 elements. Each element is 16-bit. The codebook is calculated in the PC and is converted into the 2's complement format. The data width required is 13-bit and the data is sign extended to be 16-bit before it is sent to the Block RAM.

Indices are all stored in another Block RAM. 32×32 blocks require 1024 memory entries. All the elements are initialized to be 127, which is the address in the middle of the codeword. This address then has to be shifted, which is $127 \ll 2 = 508$, to be mapped to the real address in the Block RAM that stores the codebook. The index of the upper pixel is readout, and used as the address to fetch data from the codebook. The first element of the codeword is used to calculate the absolute difference with the first coefficient, after the Hadamard transform. It is then compared to the one obtained from the left pixel. After comparison, the index with smaller distance will be saved in the current entry, which will be used as the left pixel for the next pixel. The upper pixel's index is stored in the address that could be obtained by reducing the current address by 32. After the encoding of the whole image is finished, the indices stored in the Block RAM will be readout through the USB to the host PC, where the compressed image is reconstructed.

The operating speed was set to be 80MHz after synthesis. The Hadamard transform took 16 clock cycles, and the predictor 3 cycles. The search engine takes much longer time. The rough estimation could be about 100 clock cycles. Therefore processing time for each 4 block was about $(16+3+100)/80M \approx 1.5\mu s$. For the 128×128 test image, it takes $\approx 1.536ms$. The data transmission through USB operated at the fixed rate of 8-bit@48MHz and it took $0.43\mu s$ for buffering out the indices in the Block RAM. This short period is negligible as compared with the processing time for each image.

If more resources are used, the image size and codebook size could all be

increased. The current implementation of the compression architecture is essentially a proof-of-concept. Future work could focus on integrating the pixel array and memory bank with the compression engine.

6.6 Performance Comparison

The performance comparison of the above 4 image compression algorithms in terms of PSNR, BPP, complexity is given in Table 6.9, with the results obtained from software simulation and ASIC/FPGA implementation are shown.

The 64×64 image is Toy Bear, which is the 3^{rd} image in the 3^{rd} row shown in Fig. 3.15. This image was selected for comparison because the compression system ‘Adapt-Q’ is implemented with 64×64 digital pixel array in Application Specific Integrated Circuit (ASIC) and only captured images could be compressed rather than the standard test images. The selected 512×512 *Lena* image was used to enable comparisons with other algorithms.

Table 6.9: Performance Comparison of the 4 Compression Algorithms

Compression Schemes	64×64 (Toy Bear)				512×512 (Lena)				Complexity	
	Simulation PSNR	BPP	ASIC/FPGA PSNR	BPP	Simulation PSNR	BPP	FPGA PSNR	BPP	Enc-oder	Dec-oder
Adapt-Q ¹	27.62	1.025	27.54*	1.023*	28.53	0.75	28.43	0.79	Low	Low
VPIC ²	26.1	0.9375	26.01	0.9375	28.49	0.9375	28.41	0.9375	Low	Low
CS ³	32.78	2.5	32.78	2.5	30.69	2.5	30.69	2.5	Low*	High
CSVQFS ⁴	27.71	0.5625	27.71	0.5625	29.8	0.5625	29.8	0.5625	Med.†	High

1. Adaptive Quantization with adapt- η + Hilbert Scan + DPCM + QTD

2. Visual Pattern Image Coding for TFS DPS with 2 Patterns

3. CS in both Spatial (50%) Domain and Bit (5-bit) Domain

4. CSVQ with Fast Search in Hadamard Domain and with MAE as the distance metric

*. Results obtained from ASIC. Other results in this column by default are obtained from FPGA

†. Digital Logics with no Computations

‡. Medium

For ‘Adapt-Q’ and ‘VPIC’, the simulation results are slightly better than the ASIC/FPGA results in terms of higher PSNR. This is because in ASIC/FPGA implementation, the multiplication in ‘Adapt-Q’ is replaced by shift and addition operations, and the binary coding of ‘VPIC’ is based on the timing sequence in FPGA rather than the pixel value in simulation. The BPP for ‘Adapt-Q’ is not fixed, as it is image dependent. The BPP for a small image is higher than for a large image. It is because a larger image usually has more inter-pixel redundancy,

and thus lower entropy and lower BPP. In contrast, the other three compression algorithms were designed to have fixed BPP.

For ‘CS’ and ‘CSVQFS’, the encoding stage involves no floating point arithmetic. The multiplication and square root operations in ‘CSVQFS’ are eliminated by using ‘MAE’ as the distance metric. Therefore, for these two compression algorithms, the encoding in both software simulation and FPGA implementation yields the same bit sequence and thus the same reconstruction output in the decoder.

The advantage of ‘Adapt-Q’ is that it has both low complexity encoder and decoder, and when image becomes larger and sparser, the PSNR could be increased and the BPP could be reduced. This system is suitable for focal plane compression in low power portable and/or remote sensing imaging applications. Its disadvantage is that the BPP is not fixed, making it difficult to distinguish compressed data from one frame to another.

The advantage of ‘VPIC’ is that it exploits the timing information obtained from the TFS DPS to reduce encoding complexity. The BPP is fixed for all images when a predefined bit allocation scheme is established. Its encoder and decoder are simpler than ‘Adapt-Q’. The PSNR is slightly lower than that of ‘Adapt-Q’ while the BPP is 20% higher for a 512×512 image. Therefore, the overall performance is worse than ‘Adapt-Q’, as a result of a simpler coding scheme. The fixed BPP could ease the burden in decoding the compressed bit sequence generated from the encoder. The disadvantage is that the edges are quite often poorly reconstructed. This system is also suitable for portable imaging applications performing focal plane compression.

The advantage of ‘CS’ is that there are no arithmetic operations in the encoder, where only logical operations are performed. One disadvantage is that its decoder is computationally intensive and thus it takes a longer time to reconstruct the image. However, it yields the best image quality with much higher BPP than all other 3 compression schemes. It is suitable for sensor network applications for which the complexity in the encoder is low and the changes in the scene rarely occurs. However, the power and computations in the decoder are not critical.

The ‘CS’ system directly sub-samples image in both spatial and bit domain

to achieve compression. The sub-sampled data are then used to reconstruct the original image in the decoder. Randomly selected rows from the identity matrix are used to form the sensing matrix. Therefore, no computations are required in the encoder. This system is the direct implementation of the CS framework. On the other hand, in the ‘CSVQFS’ system, CS is used to speed up a compression algorithm called VQ. In this work, CS encoding is applied to reduce the number of elements in the codebook to reduce the computations for calculating the distance measures, and thus, to accelerate the search speed. In the decoder, the quality of the coarsely reconstructed image could be improved by CS decoding.

The advantage of ‘CSVQFS’ is that it achieves reasonably high PSNR at the lowest BPP with medium encoding complexity. For 64 image, it achieves slightly higher PSNR than the ‘Adapt-Q’ with roughly half of its BPP. The disadvantage is that it requires on-chip memory to store codebook and prediction index and thus consumes more power for memory access. The decoding process of this system is also complex, while it offers the possibility to coarsely reconstruct an image with lowest complexity. If more details are needed, complex computations could be performed to get a refined image. This system is also suitable for wireless sensor network applications that occasionally capture scenes.

6.7 Summary

In this chapter, we presented how the CS framework could be integrated into the VQ compression scheme. The proposed CSVQ alleviates the computational complexity in image coding by reducing the dimensionality of VQ. A predictive partial distance search (PPDS) algorithm for fast codebook searching was proposed to boost the speed of VQ encoding. A hybrid image compression scheme combining CS and PPDS fast searching was proposed to simultaneously reduce implementation complexity and speed up the codebook search process. When the number of measurements is $m = 9$ for a 4×4 image block, the PSNR sacrifice was 0.57 dB but the number of additions, subtractions, multiplications, and square roots operations were only 3.46%, 3.28%, 3.2%, and 0.53% of that a conventional full VQ search. The proposed scheme was validated on an FPGA platform. It is well suited to wireless sensor network applications (e.g. environment monitoring)

for which scenes are captured at low frequencies.

Chapter 7

Conclusion

This thesis has investigated a number of image compression schemes and architectures suitable for integration with CMOS imagers. A novel image compression processor based on hybrid predictive boundary adaptation processing and QTD encoder was proposed. This processor was designed and successfully integrated with a 64×64 pixel sensor array in CMOS $0.35 \mu m$ technology. The whole chip occupies $3.2 \times 3.0 mm^2$ of silicon area. The power consumption reported for the compression engine was as low as 2 mW at 30 frames/s. Differential Pulse Code Modulation (DPCM) was used to exploit spatial redundancy of an image and the residual values generated were then adaptively quantized by applying the Fast Boundary Adaptation Rule (FBAR). The quantized bit stream was then further compressed by Quadrant Tree Decomposition (QTD), which was operated on-the-fly. Pixel value readout followed Hilbert scanning path. The designed sensor array was based on a Time-to-First Spike Digital Pixel Sensor architecture. Each pixel integrates 8-bit storage capacity with an embedded Static RAM, which was reused to store the quadrant tree. This saves significant power and silicon area. 26.52 dB PSNR was obtained at the compression ratio of 0.75 BPP on 256×256 resolution images. The overall performance of this integrated imaging system makes it suitable for portable and remote sensing imaging applications.

To further improve image quality, a novel image compression scheme based on the visual pattern image coding (VPIC) algorithm and optimized for Time-to-First Spike Digital Pixel Sensors was proposed. Multiplication and square root operations were eliminated and replaced by addition and shifting operations.

Each element in a block was compared with the block mean. This information was used to find out the closest visual pattern by binary correlation operation. Isometry operations were introduced in this system to expand to visual patterns of different orientations, leading to lower memory requirements. Exhaustive search and pattern re-utilization completely removed the need to evaluate gradient angle and block edge polarity. Reported image quality for the standard test image *Lena* was 29 dB at 0.875 BPP. This compression scheme is thus suitable for on-chip implementation and use in mobile imaging devices.

The two previous image compression systems were both based on the universally adopted “sample and then compress” paradigm. In this thesis, we also looked at ways to perform compression while acquiring the image. More specifically, a new image compression scheme based on compressive sampling theory was proposed. The proposed system, implemented using an FPGA platform interfaced with a CMOS imager, simultaneously samples and compresses images in both spatial and bit domains. A sensing matrix was proposed to encode the image with no multiplication operation needed. The bit domain undersampling scheme was introduced to significantly reduce the integration time for Time-to-First Spike Digital Pixel Sensors. The original image was reconstructed by l_1 -norm minimization linear programming algorithms. The hybrid system achieved 29 dB PSNR at 2 BPP for 256×256 resolution images, at no computational cost. Wireless sensor networks are a potential application of this system as most of the computations are shifted from the encoder to the decoder.

In the final part of the thesis, we applied compressive sampling to an image compression algorithm to reduce its computational complexity. More specifically, an image compression method based on vector quantization (VQ) with shrunk codeword length and reduced number of searches was developed and implemented in FPGA. Compressive Sampling was integrated into Vector Quantization to reduce the number of elements of the codeword, and thus in turn reduce the number of Euclidean Distance computations. As a consequence, about half of the multiplications and additions were eliminated. A predictive partial distance search algorithm was devised to boost the speed of the codebook search. With the above schemes, the codebook is effectively shrunk in both horizontal and vertical

directions. The system can achieve 29.2 dB at 0.5625 BPP. When the number of measurements is $m = 9$ for a 4×4 image block, the PSNR sacrifice was 0.57 dB but the number of additions, subtractions, multiplications, and square roots operations were only 3.46%, 3.28%, 3.2%, and 0.53% of that of a conventional full VQ search. This makes this scheme well suited to wireless sensor network applications (e.g. environment monitoring) for which scenes are captured at low frequencies.

Future Work

This thesis has validated a number of potential compression schemes, each offering trade-offs between silicon area, memory, operating frame rate, and power consumption. Future work could focus on:

- Investigating a faster boundary point adaptation method based on the trajectory of the boundary point so as to further enhance image quality with negligible hardware overhead.
- Investigating a charge-based analog implementation of VPIC to increase computation speed and improve image quality.
- Exploring a charge-based analog implementation of compressive sampling for dimension reduction in the CSVQ system so as to reduce silicon area.
- Applying Compressive Sampling to acquire the difference between two high speed consecutive frames and exploit sparsity of temporal information so as to reduce memory requirements for storing addresses.

List of Publications

Journal Papers:

1. Y. Wang, A. Bermak, and F. Boussaid, “Low complexity spatial and bit resolution compressive sampling hardware for CMOS Imager”, *IEEE Transactions on Circuits and Systems I*, Submitted.
2. Y. Wang, A. Bermak, and F. Boussaid, “Compressive Sampling for Dimensionality Reduction in Vector Quantization”, *IEEE Transactions on Circuits and Systems I*, Submitted.
3. S. Chen, A. Bermak, and Y. WANG, “A CMOS Image Sensor With On-Chip Image Compression Based on Predictive Boundary Adaptation and Memoryless QTD Algorithm”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19, Issue 4, pp. 538-547, April 2011.

Conference Papers:

1. Y. Wang, A. Bermak, and F. Boussaid, “Reduced Dimension Vector Quantization Encoding Method for Image Compression”, *The IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2011)*, Beirut, Lebanon, Dec 11-14, 2011
2. Y. Wang, A. Bermak, and F. Boussaid, “VPIC-based Image Compression Technique for Sensor Network Applications”, *Asia Symposium & Exhibits on Quality Electronic Design (ASQED)*, 2011, Kuala Lumpur, Malaysia, July 19-20, 2011.
3. M. Zhang, Y. Wang, and A. Bermak, “Block-Based Compressive Sampling for Digital Pixel Sensor Array”, *Asia Symposium & Exhibits on Quality*

Electronic Design (ASQED), 2010, pp. 9-12, Penang, Malaysia, Aug 3-4, 2010.

4. Y. Wang, A. Bermak, and F. Boussaid, "FPGA implementation of compressive sampling for sensor network applications", *Asia Symposium & Exhibits on Quality Electronic Design (ASQED)*, 2010, pp. 5-8, Penang, Malaysia, Aug 3-4, 2010.
5. Y. Wang, S. Chen, and A. Bermak, "Novel VLSI Implementation of Peano-Hilbert Curve Address Generator", *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2008, pp. 476-479, Seattle, USA, May 19-21, 2008.
6. Y. Wang, A. Bermak, A. Bouzerdoun, and B. Ng, "FPGA Implementation of a Predictive Vector Quantization image compression algorithm for image sensor applications", *IEEE International Symposium on Electronic Design, Test and Applications (DELTA)*, 2008, pp. 431-434, Hong Kong, Jan 23-25, 2008.
7. Y. Wang, S. Chen, and A. Bermak, "FPGA Implementation of Image Compression using DPCM and FBAR", *IEEE International Symposium on Integrated Circuits (ISIC)*, 2007, pp. 329-332, Singapore, Sept 26-29, 2007.
8. S. Chen, Y. Wang, A. Bermak, and D. Martinez, "A CMOS Image Sensor with on Chip Image Compression Based on Predictive Boundary Adaptation and QTD Algorithm", *IEEE SENSORS 2007*, pp. 531-534, Atlanta, USA, Oct 28-31, 2007.

Bibliography

- [1] 8-Megapixel camera sensor for iPhone 5, <http://www.apple.com/iphone>
- [2] 5-Megapixel camera sensor for iPad 2, <http://www.apple.com/ipad>
- [3] Castaneda, Karl, “Nintendo and PixArt Team Up”, *Nintendo World Report*, Retrieved 2007-02-24, <http://www.nintendoworldreport.com/news/11557>
- [4] Loudin J.D., Cogan S.F., Mathieson K., Sher A. and Palanker D.V., “Photodiode Circuits for Retinal Prostheses”, *IEEE Transactions on Biomedical Circuits and Systems*, No. 5, pp. 468-480, Oct. 2011.
- [5] Bharathram. P, Rajshekar. K, Saketh Paranjape, Sowmya. P, “Multi-core architecture of a digital Camera for increasing performance”, <http://software.intel.com>
- [6] QSI Scientific Cameras, <http://qsimaging.com/>
- [7] Technical Datasheet of ST Microelectronics, “1.3 Megapixel single-chip camera module”, <http://www.st.com>
- [8] TELEDYNE DALSA Corporation, <http://www.teledynedalsa.com/>
- [9] WANG X., Snoeij M.F., Rao P.R., Mierop A. and Theuwissen A.J.P., “A CMOS Image Sensor with a Buried-Channel Source Follower”, *2008 IEEE International Solid-State Circuits Conference (ISSCC 2008)*, pp. 62 - 595, Feb. 2008, San Francisco, California.
- [10] K.-Y. Min and J.-W. Chong, “A real-time jpeg encoder for 1.3 mega pixel cmos image sensor soc”, *IEEE 30th Annual Conference of Industrial Electronics Society*, vol. 3, pp. 2633-2636, 2004.

- [11] A. Bandyopadhyay, J. Lee, R. Robucci, and P. Hasler, "A 80 μ w frame 104 $\times$ 128 cmos imager front end for jpeg compression", *IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 5318-5321, 2005.
- [12] A. Olyaei and R. Genov, "Mixed-signal cmos wavelet compression imager architecture", *48th Midwest Symposium on Circuits and Systems*, vol. 2, pp. 1267-1270, 2005.
- [13] W. D. Leon-Salas, S. Balkr, K. Sayood, N. Schemm, M.W. Hoffman, "A Low-Complexity Circuit for On-Sensor Concurrent A/D Conversion and Compression", *IEEE Sensors Journal*, Vol. 7, Issue 9, pp. 1317 - 1325, 2007
- [14] Industry's Smallest CCD Pixel made by SONY, "Sony's CX-News", http://www.sony.net/Products/SC-HP/cx_news/vol60/np_icx681sqw.html
- [15] High QE CCD Sensor DataSheet, "DALSA IA-DJ High Quanta Imaging Sensor", <http://www.teledynedalsa.com/sensors/products/sensors.aspx>
- [16] Ohta J., "Smart CMOS Image Sensors and Applications", *CRC Press*, 2008
- [17] "CMOS Imagers: From Phototransduction to Image Processing", *Edited by Orly Yadid-Pecht and Ralph Etienne-Cummings*, Kluwer Academic, 2004.
- [18] E. Oda, K. Nagano, T. Tanaka, N. Mutoh, K. Orihara, "A 1920(H) $\times$ 1035(V) pixel high-definition CCD image sensor", *IEEE Journal of Solid-State Circuits*, vol. 24, No. 3, Jun 1989
- [19] Holst G. C., "CCD arrays, cameras, and displays", *JCD Publishing : SPIE Optical Engineering*, 1998
- [20] Full Frame CCD Sensor Datasheet, "DALSA FTF6146M Professional Digital Photography CCD image sensor", <http://www.teledynedalsa.com/sensors/products/sensors.aspx>
- [21] D. Litwiller, "CCD vs. CMOS: Maturing Technologies, Maturing Markets", *Photonics spectra*, Aug. 2005.
- [22] P. Clarke, "CMOS image sensor market set for steady growth", *EE Times*, May 13th, 2011

- [23] Lliana Fujimori, et al., “A 256×256 CMOS Differential Passive Pixel Imager with FPN Reduction Techniques”, *IEEE International Solid State Circuits Conference*, vol. 43, pp: 106-107, 2000.
- [24] E.R. Fossum, “Active pixel sensors: Are CCDs dinosaurs?”, *Proc. SPIE, Charged-Coupled Devices and Solid State Optical Sensors III*, vol. 1900, pp. 2-14, 1993.
- [25] B. Fowler, A. El Gamal, D.X.D. Yang, “A CMOS area image sensor with pixel-level A/D conversion”, *IEEE International Solid-State Circuits Conference, ISSCC 1994, Digest of Technical Papers*, pp. 226-227, Feb. 1994.
- [26] A. Kitchen, A. Bermak, A. Bouzerdoun, “PWM Digital Pixel Sensor Based on Asynchronous self-resetting scheme”, *IEEE Electron Device Letter*, vol. 25, no. 7, pp. 471-473, Jul. 2004.
- [27] A. Kitchen, A. Bermak, A. Bouzerdoun, “A Digital Pixel Sensor Array with Programmable Dynamic Range”, *IEEE Transactions on Electron Devices*, vol. 52, no. 12, Dec. 2005.
- [28] G. Pekhteryev, Z. Sahinoglu, P. Orlik, and G. Bhatti, “Image transmission over ieee 802.15.4 and zigbee networks”, *IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 3539-3542, 2005.
- [29] 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 one-chip cameras”, *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 2030-2041, 1997.
- [30] E. J. Tan, Z. Ignjatovic, and M. F. Bocko, “A cmos image sensor with focal plane discrete cosine transform computation”, *IEEE International Symposium on Circuits and Systems*, pp. 2395-2398, 2007.
- [31] Q. Luo, J. Harris, Z. Sahinoglu, P. Orlik, and G. Bhatti, “A novel integration of on-sensor wavelet compression for a cmos imager”, *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 325-328, 2002.

- [32] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, “Matia: a programmable $80\text{ }\mu\text{W}$ /frame cmos block matrix transform imager architecture”, *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 663-672, 2006.
- [33] W. D. Leon-Salas, S. Balkir, K. Sayood, N. Schemm, and M. W. Hoffman, “A cmos imager with focal plane compression using predictive coding”, *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 2555-2572, 2007.
- [34] A. Nilchi, J. Aziz, and R. Genov, “Focal-plane algorithmically-multiplying cmos computational image sensor”, *IEEE Journal of Solid- State Circuits*, vol. 44, pp. 1829-1839, 2009.
- [35] K. Aizawa, H. Ohno, Y. Egi, T. Hamamoto, M. Hatori, H. Maruyama, J. Yamazaki, “On sensor image compression”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, issue 3, pp. 543-548, Jun. 1997.
- [36] K. Aizawa, Y. Egi, T. Hamamoto, M. Hatori, M. Abe, H. Maruyama, H. Otake, “Computational image sensor for on sensor compression”, *IEEE Transactions on Electron Devices*, vol. 44, issue 10, pp. 1724-1730, Oct. 1997.
- [37] S. Narisawa, K. Masuda, T. Hamamoto, “High speed digital smart image sensor with image compression function”, *Proceedings of IEEE Asia-Pacific Conference on Advanced System Integrated Circuits 2004*, pp. 128- 131, Aug. 2004.
- [38] P. Lichtsteiner, C. Posch, T. Delbruck, “A 128×128 120db 30mw asynchronous vision sensor that responds to relative intensity change”, *IEEE International Solid-State Circuits Conference, ISSCC 2006*, pp. 2060-2069, 2006.
- [39] P. Lichtsteiner, C. Posch, T. Delbruck, “A 128×128 120 dB 15 s Latency Asynchronous Temporal Contrast Vision Sensor”, *IEEE Journal of Solid-State Circuits*, vol. 2, issue 2, pp. 566-576, 2008.
- [40] Y. M. Chi, U. Mallik, M. A. Clapp, E. Choi, G. Cauwenberghs, R. Etienne-Cummings, “CMOS Camera With In-Pixel Temporal Change Detection and

- ADC”, *IEEE Journal of Solid-State Circuits*, vol. 42, issue 10, pp. 2187-2196, 2007.
- [41] C. Posch, D. Matolin, R. Wohlgenannt, “A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression”, *IEEE International Solid-State Circuits Conference, 2010*, ISSCC 2010, pp. 400-401, 2010.
- [42] W. D. Leon, S. Balkir, K. Sayood and M. W. Hoffman, “Charge-based Prediction Circuits For Focal Plane Image Compression”, *IEEE International Symposium on Circuits and Systems, 2004. ISCAS 2004*, vol. 4, pp. 936-939, May 2004.
- [43] Z. Lin, M.W. Hoffman, W.D. Leon, N. Schemm, S. Balkir, “A CMOS image sensor for focal plane decomposition”, *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005*, pp. 5322-5325, May 2005.
- [44] Z. Lin, M.W. Hoffman, W.D. Leon, N. Schemm, S. Balkir, “Effects of charge-based computation non-idealities on CMOS image compression sensors”, *IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006*, May 2006.
- [45] Z. Lin, M.W. Hoffman, W.D. Leon, N. Schemm, S. Balkir, “A CMOS Front-End for a Lossy Image Compression Sensor”, *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007*, pp. 2838-2841, May 2007.
- [46] Z. Lin, M.W. Hoffman, W.D. Leon, N. Schemm, S. Balkir, “A CMOS image sensor with focal plane SPIHT image compression”, *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, pp. 2134-2137, May 2008.
- [47] S. Chen, A. Bermak, W. Yan, and D. Martinez, “Adaptive-quantization digital image sensor for low-power image compression”, *IEEE Transactions on Circuits and Systems I*, vol. 54, pp. 13-25, 2007.

- [48] A. Kitchen, A. Bermak, and A. Bouzerdoun, "A digital pixel sensor array with programmable dynamic range", *IEEE Transactions on Electron Devices*, vol. 52, pp. 2591-2601, Dec 2005.
- [49] A. Bermak, and A. Kitchen, "A Novel Adaptive Logarithmic Digital Pixel Sensor", *IEEE Photonics Technology Letters*, vol. 18, No. 20, pp. 2147-2149, Oct. 2006.
- [50] D. Martinez and M. M. V. Hulle, "Generalized boundary adaptation rule for minimizing rth power law distortion in high resolution quantization", *Neural Networks*, vol. 8, no. 6, pp. 891-900, 1995.
- [51] E. Artyomov, Y. Rivenson, G. Levi, and O. Yadid-Pecht, "Morton (z) scan based real-time variable resolution cmos image sensor", *IEEE Transactions on Circuits and Systems I*, vol. 15, pp. 947-952, 2005.
- [52] Delft University of Technology, VcDemo software, <http://ict.ewi.tudelft.nl>.
- [53] The USC-SIPI Image Database, <http://sipi.usc.edu/database/index.html>
- [54] E. Artyomov and O. Yadid-Pecht, "Adaptive multiple-resolution cmos active pixel sensor", *IEEE Transactions on Circuits and Systems I*, vol. 53, pp. 2178-2186, 2006.
- [55] Z. Lin, M.W. Hoffman, W.D. Leon, N. Schemm, S. Balkir, "A CMOS Image Sensor for Multi-Level Focal Plane Image Decomposition", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, No. 9, pp. 2561-2572, Oct 2008.
- [56] Chi Y.M., Abbas A., Chakrabartty S., Cauwenberghs G., "An Active Pixel CMOS Separable Transform Image Sensor", *IEEE International Symposium on Circuits and Systems, ISCAS 2009*, pp. 1281-1284, June 2009, Taipei.
- [57] Chen S., Bermak A., Wang Y., Martinez D., "A CMOS Image Sensor With On-Chip Image Compression Based on Predictive Boundary Adaptation and Memoryless QTD Algorithm", *IEEE Transactions on Very Large Scale Integration Systems*, vol. 19, No. 4, pp. 538 - 547, 2011.

- [58] Chi Y.M., Etienne-Cummings R., Cauwenberghs G., “Image sensor with focal plane change event driven video compression”, *IEEE International Symposium on Circuits and Systems, ISCAS 2008*, pp. 1862 - 1865, May 2008, Seattle, USA.
- [59] Chen D., Bovik A. G., “Visual Pattern Image Coding”, *IEEE Transactions on Communication*, vol. 38, No. 12, pp. 2137 - 2146, 1990.
- [60] Ramamurthi B., Gersho A., “Classified Vector Quantization of Images”, *IEEE Transactions on Communication*, vol. COM-34, No. 11, pp. 1105 - 1115, 1986.
- [61] Masselos K., Merakos P., Stouraitis T., Goutis C.E., “Novel Vector Quantization Based Algorithms for Low-Power Image Coding and Decoding”, *IEEE Transactions on Circuits and Systems — II: Analog and Digital Signal Processing*, vol. 46, No. 2, pp. 193 - 198, 1999.
- [62] Chen, S., Bermak, A., Wang, Y., Martinez, D., “A CMOS Image Sensor with on Chip Image Compression based on Predictive Boundary Adaptation and QTD Algorithm”, *IEEE Sensors 2007*, pp. 531 - 534, 28-31 Oct. 2007.
- [63] Candes E.J., Wakin M.B., “An Introduction To Compressive Sampling [A sensing/sampling paradigm that goes against the common knowledge in data acquisition]”, *IEEE Signal Processing Magazine* vol. 25, No. 2, pp. 21-30, March 2008.
- [64] Kawahito et al., “CMOS Image Sensor with Analog 2-D DCT-Based Compression Circuits”, *IEEE Journal of Solid-State Circuits*, vol. 32, No. 12, pp. 2029-2039, December 1997.
- [65] Vishwanath, M., Owens, R.M., Irwin, M.J., “VLSI architectures for the discrete wavelet transform”, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, No. 5, pp. 305-316, May 1995.
- [66] Goyal, V.K., Fletcher, A.K., Rangan, S., “Compressive Sampling and Lossy Compression [Do random measurements provide an efficient method of rep-

- resenting sparse signals?]", *IEEE Signal Processing Magazine*, vol. 25, No. 2, pp. 48-56, Mar 2008.
- [67] Duarte, M.R., Wakin, M.B., Baron, D., and Baraniuk, R.G., "Universal Distributed Sensing via Random Projections", *The Fifth International Conference on Information Processing in Sensor Networks, IPSN 2006*, pp. 177-185, April 19-21 2006.
- [68] Duarte, M.F., Davenport, M.A., Takbar, D., Laska, J.N., Sun, T., Kelly, K.F., Baraniuk, R.G., "Single-Pixel Imaging via Compressive Sampling [Building simpler, smaller, and less-expensive digital cameras]", *IEEE Signal Processing Magazine*, vol. 25, No. 2, pp.89-91, March 2008
- [69] Coifman, R., Geshwind, F. , and Meyer Y., "Noiselets", *Applied and Computational Harmonic Analysis*, vol. 10, no. 1, pp. 27-44, doi:10.1006/acha.2000.0313, available online at <http://www.idealibrary.com>
- [70] Lustig M., Donoho D.L., Santos J.M., Pauly J.M., "Compressed Sensing MRI", *IEEE Signal Processing Magazine*, vol. 25, No. 2, pp. 72-82, March 2008
- [71] Tropp, J., and Gilbert, A., "Signal recovery from random measurements via orthogonal matching pursuit", *IEEE Trans. on Information Theory*, 53(12), pp. 4655-4666, Dec. 2007
- [72] Figueiredo, M. A. T., Nowak, R. D., and Wright, S. J. "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems", *IEEE Journal of Selected Topics in Signal Processing: Special Issue on Convex Optimization Methods for Signal Processing*, 1(4), pp. 586-598, 2007
- [73] Ma, J., "Compressed sensing by inverse scale space and curvelet thresholding", *Applied Mathematics and Computation*, 206, pp. 980-988, 2008
- [74] Candes, E., Romberg, J., and Tao, T., "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information", *IEEE Trans. on Information Theory*, 52(2), pp. 489-509, Feb. 2006

- [75] Candes E.J., Romberg J., “Sparsity and Incoherence in Compressive Sampling”, *Inverse Problems*, vol. 23, No. 3, pp. 969-985, 2007
- [76] Candes E., Tao T., “Decoding by Linear Programming”, *IEEE Transactions on Information Theory*, vol. 51, No. 12, pp. 4203-4215, Dec. 2005.
- [77] Laska, J., Boufounos, P., and Baraniuk, R., “Finite range scalar quantization for compressive sensing”, *International Conference on Sampling Theory and Applications, SampTA*, May 2009
- [78] Laska J.N. , Boufounos P.T. , Davenport M.A. , and Baraniuk R.G., “Democracy in action: Quantization, saturation, and compressive sensing”, submitted to *IEEE Transactions on Signal Processing*, July 2009.
- [79] Candes E.J., “l1-Magic”, <http://www.acm.caltech.edu/l1magic/>
- [80] Gehm M. E., John R., Brady D. J., Willett R. M. and Schultz T. J., “Single-shot compressive spectral imaging with a dual-disperser architecture”, *Opt. Express*, 15(21), 14013-14027, 2007.
- [81] Wagadarikar A., John R., Willett R., and Brady D., “Single disperser design for coded aperture snapshot spectral imaging”, *Appl. Opt.*, 47(10), B44-B51, 2008.
- [82] Maggioni M., Davis G. L., Warner F. J., Geshwind F. B., Coppi A. C., DeVerse R. A. and Coifman R. R., “Hyperspectral microscopic analysis of normal, benign and carcinoma microarray tissue sections”, *Proc. SPIE* 6091, 2006.
- [83] Gu J., Nayar S. K., Grinspun E., Belhumeur P. N. and Ramamoorthi R., “Compressive Structured Light for Recovering Inhomogeneous Participating Media”, *European Conference on Computer Vision (ECCV)*, Marseille-France, 2008.
- [84] Shankar M., Willett R., Pitsianis N., Schulz T., Gibbons R., Kolste R. T., J. Carriere, C. Chen, D. Prather, and D. Brady, “Thin infrared imaging systems through multichannel sampling”, *Appl. Opt.*, 47(10), B1-B10, 2008.

- [85] Coskun A. F., Sencan I., Su T.-W., and Ozcan A., “Lensless widefield fluorescent imaging on a chip using compressive decoding of sparse objects”, *Opt. Express*, 18(10), 10510-10523, 2010.
- [86] Potter L. C., Ertin E., Parker J. T. and M. Cetin, “Sparsity and compressed sensing in radar imaging”, *Proc. IEEE*, 98(6), 1006-1020, 2010.
- [87] Gurbuz A. C., McClellan J. H. and Scott W. R., “A compressive sensing data acquisition and imaging method for stepped frequency GPRs”, *IEEE Trans. Signal Process.*, 57, 2640-2650, 2009.
- [88] Bobin J., Starck J.-L. and Ottensamer R., “Compressed sensing in astronomy”, *IEEE J. Sel. Top. Signal Process.*, 2, 718-726, 2008.
- [89] Mohtashemi M., Smith H., Walburger D., Sutton F. and Diggans J., “Sparse sensing DNA microarray-based biosensor: Is it feasible?”, *2010 IEEE Sensors Applications Symposium*, Limerick, Ireland, pp. 127-130, 2010.
- [90] Sheikh M., Milenkovic O. and Baraniuk R., “Designing compressive sensing DNA microarrays”, in *2nd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 141-144, 2007, St. Thomas, VI, USA
- [91] Lustig M., Donoho D. and Pauly J.M., “Sparse MRI: The application of compressed sensing for rapid MR imaging”, *Magn. Reson. Med.*, 58, 1182-1195, 2007.
- [92] Ye P., Paredes J. L., Arce G., Wu, Y. Chen C. and Prather D., “Compressive confocal microscopy”, in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, pp. 429-432, 2009.
- [93] Healy D., Brady D.J., “Compression at the Physical Interface [The A-to-I and MONTAGE programs]”, *IEEE Signal Processing Magazine*, vol. 25, No. 2, pp. 67-71, March 2008
- [94] Haupt J., Bajwa W.U., Rabbat M., Nowak R., “Compressed Sensing for Networked Data [A different approach to decentralized compression]”, *IEEE Signal Processing Magazine*, vol. 25, No. 2, pp. 92-101, March 2008

- [95] Robucci R., Leung K. C., Gray J., Romberg J., Hasler P., Anderson D.,
“Compressive sensing on a CMOS separable transform image sensor”, *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*, pp. 5125 - 5128, March 31 - April 4, 2008, Las Vegas, Nevada, U.S.A.
- [96] Robucci R. , Gray J. D. , Chiu L. K., Romberg J. and Hasler P., “Compressive sensing on a CMOS separable-transform image sensor”, *Proc. IEEE*, 98(6), 1089-1101, 2010.
- [97] Jacques L., Vandergheynst P., Bibet A., Majidzadeh V., Schmid A., and Leblebici Y., “CMOS compressed imaging by random convolution”, *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1113 - 1116, 19 -24 April 2009, Taipei, Taiwan.
- [98] Majidzadeh V., Jacques L., Schmid A., Vandergheynst P. and Leblebici Y.,
“A 256×256 pixel 76.7mW CMOS imager/compressor based on real-time in-pixel compressive sensing”, in *IEEE International Symposium on Circuits and Systems*, pp. 2956- 2959, 2010, Paris, France
- [99] Marcia R. F., Harmany Z. T., and Willett R. M., “Compressive coded aperture imaging”, in *Proceedings of the 2009 IS&T/SPIE Electronic Imaging: Computational Imaging VII*, San Jose, CA, 2009.
- [100] J. Romberg, “Compressive sampling by random convolution”, *SIAM J. Imaging Sci.*, 2(4), 1098-1128, 2009.
- [101] Raginsky M., Willett R. M., Harmany Z. T. and Marcia R. F., “Compressed sensing performance bounds under Poisson noise”, *IEEE Trans. Signal Process.*, 58(8), 3990-4002, 2010.
- [102] Raginsky M., Jafarpour S., Harmany Z., Marcia R., Willett R. and Calderbank R., “Performance bounds for expander-based compressed sensing in Poisson noise”, accepted to *IEEE Transactions on Signal Processing*, 2011.

- [103] Figueiredo M. A. T., Nowak R. D. and Wright S. J., “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems”, *IEEE J. Sel. Top. Signal Process.*, 1(4), 586-597, 2007.
- [104] Van den Berg E. and Friedlander M. P., “Probing the pareto frontier for basis pursuit solutions”, *SIAM J. Sci. Comput.*, 31(2), 890-912, 2008.
- [105] Wright S., Nowak R. and Figueiredo M., “Sparse reconstruction by separable approximation”, *IEEE Trans. Signal Process.*, 57, 2479-2493, 2009.
- [106] Daubechies I., Defrise M. and De Mol C., “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint”, *Commun. Pure Appl. Math. LVII*, 1413-1457, 2004.
- [107] Bioucas-Dias J. M. and Figueiredo M. A., “A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration”, *IEEE Trans. Image Process.*, 16(12), 2992-3004, 2007.
- [108] Davis G., Mallat S. and Avellaneda M., “Greedy adaptive approximation”, *J. Constructive Approximation*, 13, 57-98, 1997.
- [109] Donoho D. L., Elad M. and Temlyakov V. N., “Stable recovery of sparse overcomplete representations in the presence of noise”, *IEEE Trans. Inf. Theory*, 52(1), 6-18, 2006.
- [110] Tropp J. A., “Greed is good: Algorithmic results for sparse approximation”, *IEEE Trans. Inf. Theory*, 50, 2231-2242, 2004.
- [111] Dai, W., Pham, H. V., Milenkovic, O., “Quantized Compressive Sensing”, (Part of the material in this paper was submitted to the *IEEE Information Theory Workshop (ITW) 2009*, and the *IEEE International Symposium on Information Theory (ISIT) 2009*, Manuscript available online at http://arxiv.org/PS_cache/arxiv/pdf/0901/0901.0749v2.pdf
- [112] Zymnis, A., Boyd, S., and Candes, E., “Compressed Sensing with Quantized Measurements”, Manuscript available online at http://stanford.edu/~boyd/papers/quant_compr_sens.html

- [113] Boufounos, P., and Baraniuk, R. G., “Quantization of sparse representations”, *Rice ECE Department Technical Report TREE 0701 - Summary appears in Data Compression Conference (DCC)*, Snowbird, Utah, March 2007
- [114] Boufounos, P., and Baraniuk, R. G., “1-Bit compressive sensing”, *Conf. on Info. Sciences and Systems (CISS)*, Princeton, New Jersey, March 2008
- [115] Chen, S., Bermak, A., Wang, Y., Martinez, D., “A CMOS Image Sensor with on Chip Image Compression based on Predictive Boundary Adaptation and QTD Algorithm”, *IEEE Sensors 2007*, pp. 531-534, 28-31 Oct. 2007.
- [116] Haupt, J., Nowak, R., “Signal Reconstruction From Noisy Random Projections”, *IEEE Transactions on Information Theory*, vol. 52, No. 9, pp. 4036 - 4048, Sept. 2006
- [117] Khalid S., “Introduction to data compression”, 3rd ed, Amsterdam : Elsevier, c2006
- [118] Y. Linde, A. Buzo, and R.M. Gray, “An Introduction for Vector Quantization Design”, *IEEE Transactions on Communications*, COM-28, pp. 84-95, Jan, 1980
- [119] Bei, C. D. and Gray, R. M., “An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization”, *IEEE Transactions on Communications*, vol. COM-3, No. 10, pp. 1132 - 1133, Oct 1985
- [120] Fissore, L., Laface, P., Massafra, P., and Ravera, F., “Analysis and improvement of the partial distance search algorithm”, *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993. ICASSP-93.*, vol. 2, pp. 315-318, Minneapolis, MN, April 27-30, 1993
- [121] Ndifor, J. N. and Ellis, T., “Predictive Partial Search Algorithm for Vector Quantization”, *Electronics Letters*, vol. 27, No. 19, pp. 1722-1723, Sept 1991
- [122] Li, W. and Salari, E., “A fast vector quantization encoding method for image compression”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, No. 2, pp.:119-123, Apr 1995

- [123] Chen, S. X. and Li, F. W., “Initial Codebook Method of Vector Quantization in Hadamard Domain”, *Electronics Letters*, vol. 46, No. 9, pp. 630-631, April 2010
- [124] Lu, Z. M., Pan, J. S., and Sun S.H., “Efficient Codeword Search algorithm based on Hadamard Transform”, *Electronics Letters*, vol. 36, No. 16, pp. 1364-1365, Aug 2000
- [125] Shen, G. B., Liou, M. L., “An efficient codebook post-processing technique and a window-based fast-search algorithm for image vector quantization”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, No. 6, pp. 990 - 997, Sept 2000
- [126] M. Davenport, M. Duarte, M. Wakin, J. Laska, D. Takhar, K. Kelly, and R. Baraniuk, “The smashed filter for compressive classification and target recognition”, *Computational Imaging V., Proceedings of the SPIE*, vol. 6498, pp. 64980H (2007), San Jose, California, January 2007
- [127] Online available at <http://www.vlsiip.com/intel/vhdlf.html>

Glossary

ADC	Analog-to-Digital Converter
AER	Address-Event Representation
AGC	Automatic Gain Control
APS	Active Pixel Sensor
AR	Asynchronous Reset
ASIC	Application-Specific Integrated Circuit
BP	Boundary Points
BPP	Bit-Per-Pixel
CCD	Charge-Coupled Device
CDS	Correlated Double Sampling
CFA	Color Filter Array
CIS	CMOS Image Sensor
CMOS	Complementary Metal-Oxide-Semiconductor
CO	Convex Optimization
CS	Compressive Sampling / Compressed Sensing
CSLC	Compressive Structured Light Code
CSVQ	Compressively Sampled Vector Quantization
CVQ	Classified Vector Quantization
DC	Direct Current
DCT	Discrete Cosine Transform
DMD	Digital Micromirror Device
DNA	Deoxyribonucleic Acid
DPCM	Differential Pulse Code Modulation
DPS	Digital Pixel Sensor
DSP	Digital Signal Processor
DST	Discrete Sine Transform
DWT	Digital Wavelet Transform
FBAR	Fast Boundary Adaptation Rule
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
FPS	Frame-Per-Second
I-V	Current-to-Voltage
JPEG	Joint Photographic Experts Group
LBG	Linde-Buzo-Gray
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bits
LUT	Look-Up Table

MADC	Multiplying Analog to Digital Converter
MAE	Mean Absolute Error
MATIA	MAtrix Transform Imager Architecture
MOS	Metal Oxide Semiconductor
MP	Matching Pursuit
MPEG	Moving Picture Experts Group
MRI	Magnetic Resonance Imaging
MSB	Most Significant Bits
MSE	Mean Square Error
NMOS	N-channel MOS Field Effect Transistor
Op-Amp	Operational Amplifier
PC	Personal Computer
PD	Photodiode
PDS	Partial Distance Search
PPDS	Predictive Partial Distance Search
PPS	Passive Pixel Sensor
PRNG	Pseudo-Random Number Generator
PSNR	Peak Signal-to-Noise Ratio
PWM	Pulse Width Modulated
QE	Quantum Efficiency
QTD	Quadrant Tree Decomposition
QVGA	Quarter Video Graphics Array
RAM	Random Access Memory
RIP	Restricted Isometry Property
RMS	Root Mean Square
SI	Start Integration
SNR	Signal-to-Noise Ratio
SPIHT	Set Partitioning In Hierarchical Trees
SR	Spectral Response
SRAM	Static Random Access Memory
SR Latch	Set-Reset Latch
SSE	Sum of Squared Error
STD	Standard Deviation
TFS	Time-to-First Spike
TV	Total Variation
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VLC	Variable Length Coding
VLSI	Very-Large-Scale Integration
VMM	Vector Matrix Multiplier
VPIC	Visual Pattern Image Coding
VQ	Vector Quantization
WT	Wavelet Transform