CMOS IMAGE SENSORS WITH COMPRESSIVE SENSING ACQUISITION

CMOS IMAGE SENSORS WITH COMPRESSIVE SENSING ACQUISITION

BY

MOHAMMADREZA DADKHAH

B.Sc., Isfahan University of Technology-Isfahan-Iran

M.Sc., Isfahan University of Technology-Isfahan-Iran

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF McMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2013) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	CMOS IMAGE SENSORS WITH COMPRESSIVE SENSING ACQUISITION
AUTHOR:	Mohammadreza Dadkhah B.Sc., Isfahan University of Technology-Isfahan-Iran M.Sc., Isfahan University of Technology-Isfahan-Iran
SUPERVISOR:	Prof. Shahram Shirani and Prof. M. Jamal Deen
NUMBER OF PAGES:	181

This thesis is dedicated to my parents for supporting me with affection and love.

Abstract

The compressive sensing (CS) paradigm provides an efficient image acquisition technique through simultaneous sensing and compression. Since the imaging philosophy in CS imagers is different from conventional imaging systems, new physical structures are required to design cameras suitable for CS imaging.

While this work is focused on the hardware implementation of CS encoding for CMOS sensors, the image reconstruction problem of CS is also studied. The energy compaction properties of the image in different domains are exploited to modify conventional reconstruction problems. Experimental results show that the modified methods outperform the 1-norm and TV (total variation) reconstruction algorithms by up to 2.5dB in PSNR.

Also, we have designed, fabricated and measured the performance of two real-time and area-efficient implementations of the CS encoding for CMOS imagers. In the first implementation, the idea of active pixel sensor (APS) with an integrator and in-pixel current switches are used to develop a compact, current-mode implementation of CS encoding in analog domain. In another implementation, the conventional three-transistor APS structure and switched capacitor (SC) circuits are exploited to develop the analog, voltage-mode implementation of the CS encoding. With the analog and block-based implementation, the sensing and encoding are performed in the same time interval, thus making a real-time encoding process. The proposed structures are designed and fabricated in 130nm technology. The experimental results confirm the scalability, the functionality of the block read-out, and the validity of the design in making monotonic and appropriate CS measurements.

This work also discusses the CS-CMOS sensors for high frame rate CS video coding. The method of multiple-camera with coded exposure video coding is discussed and a new pixel and array structure for hardware implementation of the method is presented.

Acknowledgements

I would like to express my deep appreciation and sincere thanks to my supervisor Prof. Shahram Shirani for his guidance, efforts, patience and encouragement. All he has ever taught me will surely outlive this phase of my life.

I would also like to express my gratitude to my other supervisor Prof. M. Jamal Deen. It is difficult to put in words the extent of my appreciation for him. He has helped, supported and encouraged me throughout this work. His dedicated guidance and unconditional assistance have motivated me in the past four years to become a better scholar.

I wish to thank Prof. S. Kumar and Prof. N. Nicolici of the ECE department for agreeing to be members of my supervisory committee and their advices.

My thanks also go to my colleagues at the Multimedia Communications and the Nanoelectronics and Optoelectronics Research Labs, D. Palubiak, M. Eldesouki, R. Pournaghi, H. Kasiri, M. Nabaee, and A. Behnad. I also thank Dr. O. Marinov for his technical assistance.

I thank the CMC Microsystems for arranging the fabrication of the test chips. I would also like to thank the administrative and technical staff of the ECE department of McMaster University: Cheryl Gies, Helen Jachna, Tyler Ackland, Terry Greenlay, Cosmin Coroiu, and Dan Manolescu.

Last but certainly not least, I thank my parents and my lovely sisters for their unconditional support and encouragement.

Notation and abbreviations

ADC	Analog-to-Digital Converter
AL	Array Level
APS	Active Pixel Sensor
ASMP	Adaptive Sparsity Matching Pursuit
BB	Block-by-Block
BP	Basic Pursuit
CB	Column-of-Block
CDS	Correlated Double Sampling
CMOS	Complementary Metal-Oxide Semiconductor
CoSaMP	Compressive Sampling Matching Pursuit
CS	Compressive Sensing
CT	Computed Tomography
DCT	Discrete Cosine Transform
DMD	Digital Micro-mirror Device
DPS	Digital Pixel Sensors
DWT	Discrete Wavelet Transform
FI	Fluorescence Imaging
FL	Frame Level

FLIM	Fluorescence Lifetime Imaging
FSM	Finite State Machine
GI	Gastrointestinal
LFSR	Linear Feedback Shift Register
LP	Laplacian Pyramid
MRI	Magnetic Resonance Imaging
nMOS	n-channel Metal-Oxide-Semiconductor
OMP	Orthogonal Matching Pursuit
Op-Amp	Operational Amplifier
PBP	Pixel-by-Pixel
PCB	Printed Circuit Board
PL	Pixel Level
pMOS	p-channel Metal-Oxide-Semiconductor
PPS	Passive Pixel Sensor
PSNR	Peak Signal-to-Noise Ratio
RIP	Restricted Isometry Property
ROI	Region Of Interest
ROMP	Regularized Orthogonal Matching Pursuit
SC	Switched Capacitor
SNR	Signal-to-Noise Ratio
SOCP	Second Order Cone Programming
SP	Subspace Pursuit
StOMP	Stagewise Orthogonal Matching Pursuit
TV	Total Variation
TIA	Trans-Impedance Amplifier

VLSI	Very-Large-Scale Integration
VMM	Vector-Matrix Multiplier

Contents

List of Figures

Chapter 1

Introduction and applications

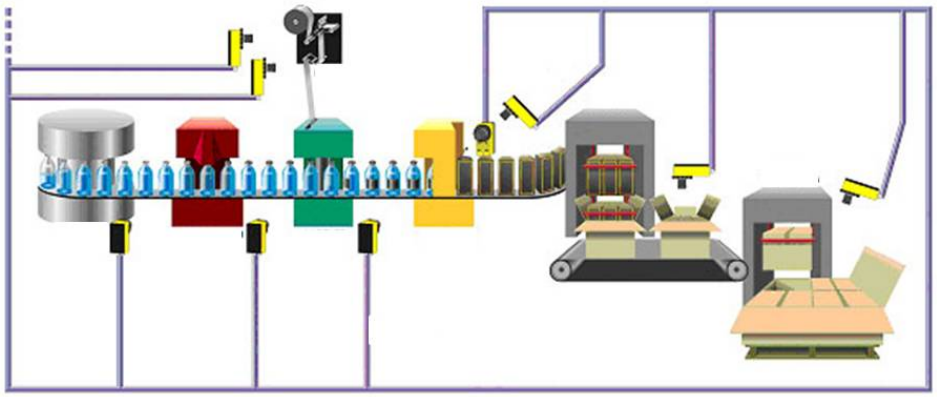
1.1 Applications of CMOS image sensors

The advances in CMOS technology have paved the way for the development of smart imaging systems. Therefore, in recent years, CMOS image sensors have found extensive applications in hand-held digital cameras as well as cameras in cell phones, computers, biomedical imaging, and smart monitoring systems. There are specific requirements and restrictions for each applications using imaging systems. The sensitivity of the image sensor for low-level-light applications, low power consumption, low cost, high speed, high resolution imaging and integrability of imaging systems with processing circuitry are some of the required characteristics for different applications. In this section, some applications of image sensors that consider their important performance requirements are introduced.

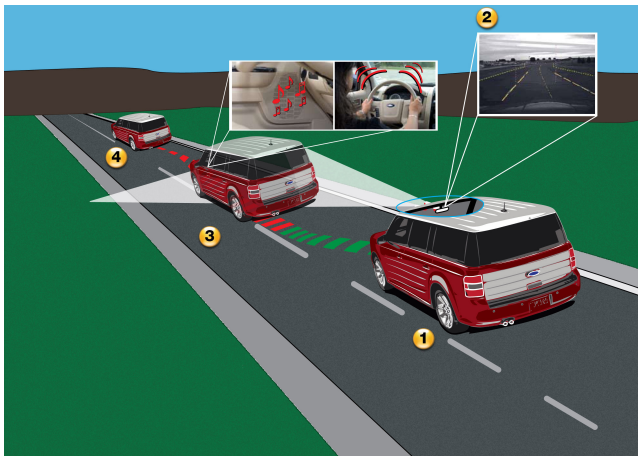
1.1.1 Machine vision

Machine vision systems are used in imaging-based automatic inspection and industrial analysis [1]. Some of the main applications of machine vision are face recognition, object tracking, printed circuit board (PCB) or semiconductor inspection, traffic monitoring systems, lane departure warning systems, automatic cruise control, etc, [2]. In most of these applications, some specific features in the captured image are important instead of the entire image. Therefore, image processing and feature extraction algorithms are performed to find some particular information from the visual data captured by the camera. Since, the extracted information is used to control automatic tasks or make critical decisions, then fast and reliable interpretation of the visual data is very important. For example, image sensors with fast and accurate imaging capability and good performance in different temperature and light conditions are required in machine vision systems.

Image processing is a critical part of an entire vision system. Therefore, the processing units should be fast to detect the desired features and make the required decisions in an appropriate time. However, the imaging and processing can be cointegrated in the same chip to improve the overall speed of the system. Therefore, fast and reliable image sensors with the capability of on-chip signal processing or feature extraction during the acquisition can be helpful in machine vision applications. Figure 1.1 shows two applications of machine vision systems in industrial production line inspection, manufacturing and quality control, and a lane departure warning to reduce the risk of car accident.



(a)



(b)

Figure 1.1: (a) Machine vision system for different parts of an industrial production line [3]. (b) Lane departure warning system [4].

1.1.2 Sensor networks

Sensor networks perform parallel information extraction using a distributed array of sensors. Visual (vision-based) sensor networks use several cameras to collect the visual information of a scene from different viewpoints. Some of the applications of visual sensor

networks are environmental monitoring, surveillance, and smart homes, etc, [5]. Figure 1.2, shows the distribution of several cameras and their communication links for monitoring in a military application. Each node in a visual sensor network should be capable of extracting, processing and transmitting the visual data. Therefore, a node consists of an image sensor, an embedded processor and a wireless transceiver. Since the sensor nodes are usually battery-operated, then the power consumption is one of the main concerns in the design of different parts of the node.

In some applications, the entire sensing process should be real-time, thus requiring fast and less complex imaging and processing units. Therefore, each node should extract only useful and new data from the scene. Also, redundant information should be locally removed at each node to improve the efficiency and reduce the amount of the transmitted data. This leads to the use of local image processing and data compression algorithms. Therefore, low cost, low power and smart image sensor structures which are compatible with on-chip signal processing are required in visual sensor network applications [6].

1.1.3 Endoscopic capsules

Various imaging techniques are used for gastrointestinal (GI) examination. For example, in conventional endoscopic methods, a tube of fiber optics is placed inside the GI tract of the patient to provide direct visualization of the GI tract. The movement of the endoscope can be controlled from outside the body to capture a visible image from the desired area for medical diagnosis. However, the conventional endoscopic examinations are invasive and hurtful for the patient.

Endoscopic capsules can be used as a non-invasive and pain-free method for GI video capture [8–10]. The capsule is a pill-sized video camera which travels through the GI tract of

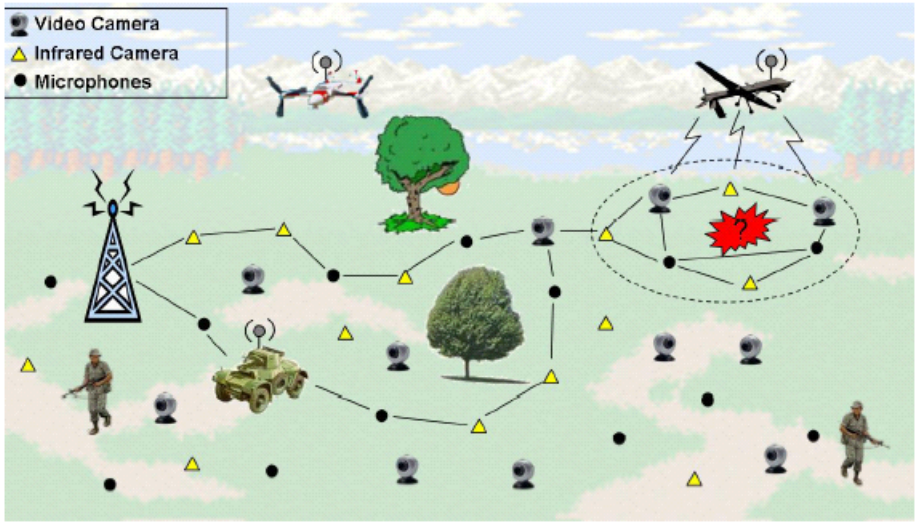
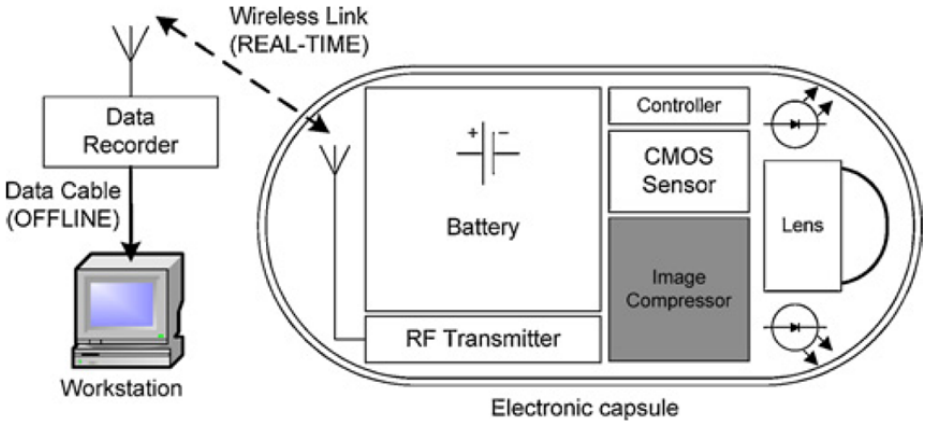


Figure 1.2: Distribution of different cameras in the development of a visual sensor network for military applications [7].

the patient for around 8-10 hours [11]. The captured video (or a sequence of images) during this time is sent outside the body and received by the recording electronic system connected to the patient waist. The recorded information is downloaded from the electronic system to reconstruct the image for medical diagnosis. However, the capsule should be swallowed by the patient and is supposed to easily move inside the GI tract. This makes some restrictions on the size of the capsule. Also, there should be enough power for imaging, processing and transmitting the video data during the entire recording time. Therefore, the main concerns in designing the endoscopic capsules are size and power consumption. To efficiently use the power budget of the system, image processing and compression algorithms can be used to process and reduce the data before the transmission. Also, the processing and imaging parts can be cointegrated to optimize the size of the capsule. Therefore, image sensors with high quality, small size, and low power consumption with high level of integrability are demanded in endoscopic capsule design.

Figure 1.3 shows a schematic diagram of an endoscopic capsule with embedded image compressor and some sample image presented in [11]. As shown, the relative size of the battery compared to other units, indicates the importance of the power consumption even in terms of the size of the capsule.



(a)

(b)

Figure 1.3: (a) General schematic and (b) Sample images of the endoscopic capsule presented in [11].

1.1.4 DNA microarrays

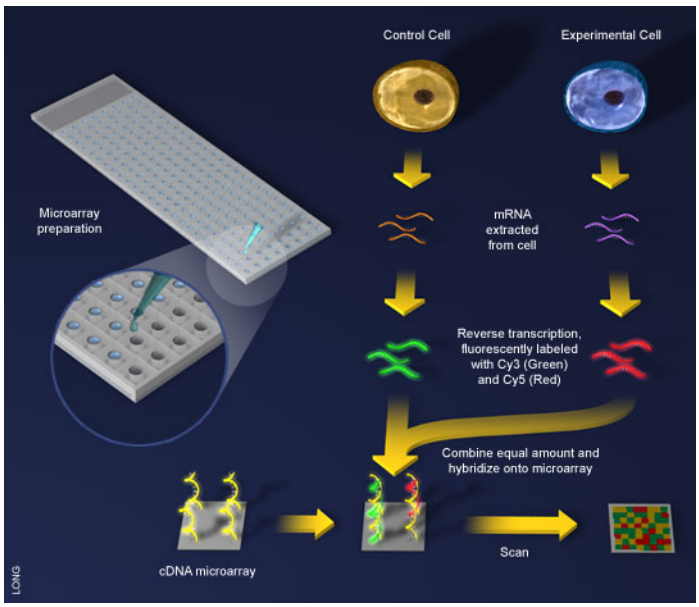
The characteristics of living creatures are expressed by their genes. Also, the genetic structure of each species is encoded in its DNA structure. Each type of cells can be specified by its DNA sequence as its biological signature. Therefore, examination of DNA can be used to describe dissimilarities between species and recognize differences in gene expressions caused by diseases.

The presence of different DNA types should be investigated in identification of the biological samples. However, they are a huge number of DNA strains which should be examined. Therefore, DNA microarrays are used for parallel examination of a large number of DNA strains at the same time [12–14]. A DNA microarray is an array of genetic sensors. Each sensor or spot on the array contains a large number of DNA sequences called probes. Each probe can identify only its complementary DNA sequence. To test the biological sample for its DNA sequences, the test samples or targets are labeled by a florescence tag to make them detectable. The DNA microarray is exposed to the sample and each DNA target existing in the sample bonds to its complementary probe during the process termed hybridization. The DNA microarray is then washed up to remove the unbonded sequences. To detect the bonded DNA sequences, the microarray is illuminated and scanned to make an image from the microarray. The level of the fluorescent reflection from each spot shows the level of gene expression for the DNA sequence corresponding to that spot. An image sensor with large number of sensors is required to measure the reflection from the different spots.
Figure1.4(a) shows a general schematic of using a DNA microarray to study the genetic differences between two samples called a control cell and an experimental cell. By extracting mRNA strains and through the reverse-transcription process, the complementary DNA (cDNA) for each sample is obtained. The cDNA molecules are labeled by different florescent molecules in red and green and exposed to the microarray. Then, the microarray is scanned to obtain the color image shown in Figure 1.4(b).

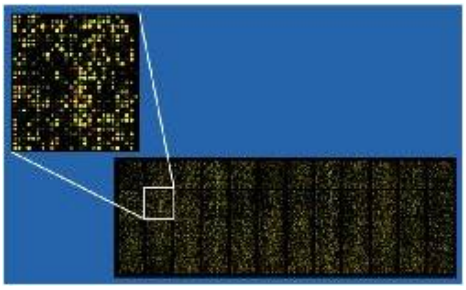
1.1.5 Fluorescence imaging

As an optical imaging method, fluorescence imaging (FI) is extensively used in biomedical applications such as DNA microarrays scanning, cancer diagnostics, identification of tumor

boundaries, blood and lymph vessels assessments, etc [15]. The information obtained from optical imaging are very useful because of different absorbtion rate of the light for different wavelengths [16]. Also, fluorescence imaging is noninvasive compared with other medical imaging methods such as computed tomography (CT), X-ray, and magnetic resonance imaging (MRI) [17].



(a)



(b)

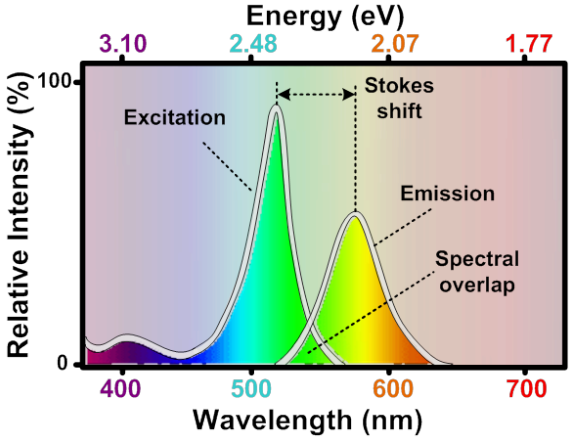
Figure 1.4: (a) General schematic for genetic evaluation of two biologic samples using DNA microarray [18], and (b) The resulted image from the output of the microarray [19].

The principle of FI is based on the fluorescence property of the certain molecules called fluorophores, fluorochromes, or fluorescent dyes [20]. When these molecules are illuminated at some particular wavelengths, their energy level rises to a new excited state. Then, as the energy level decreases, they start emitting light at longer wavelengths. The total emission time is called the fluorescence lifetime. Figure 1.5(a) shows the excitation and emission intensity pulses as a function of wavelength. Using the information extracted from the intensity and wavelength of the emitted light for different areas of the tissue, a detailed, high-contrast image can be obtained.

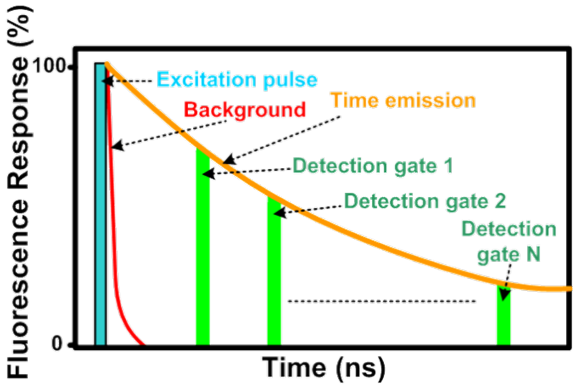
However, sometimes different molecules have overlapping fluorescence spectra, thus making them difficult to distinguish. In this case, fluorescence lifetime imaging (FLIM) as shown in Figure 1.5(b) can be used to solve the problem of overlapped spectra. FLIM uses the decay time of the fluorescence response in time domain, i.e. fluorescence lifetime, as the measure of evaluation [17]. The lifetime can be calculated by multiple sampling of the response in several detection intervals, i.e. detection gates in Figure 1.5(b). Since the intensity decreases exponentially and lifetime is very short, then the imaging system should be fast to capture several samples during the lifetime. Also, the intensity could be very small, especially at the end of the lifetime interval. Therefore, a sensitive and high-frame rate image sensor is required to provide accurate values of the FLIM response.

1.2 CMOS imagers - potential and features

CMOS image sensors use photodiodes and some peripheral electronics fabricated in standard CMOS technologies for image acquisition and processing. Photodiodes convert the photons to electrons, then the electric signals are used to represent, process and display the image information. The image sensor includes an array of sensors while each sensor



(a)



(b)

Figure 1.5: (a) Excitation and emission pulses vs. wavelength, and (b) Fluorescence response in time-domain and fluorescence lifetime measurements [17].

represents a pixel of the image which sends output to the level of the light for a specific area in the scene. There are different requirements in designing an image sensor which are progressively fulfilled by the advances in CMOS technology. Some of the interesting features provided by CMOS technologies for imaging systems are the following.

Small size. The size of each sensor in the array, i.e. pixel size, is important in providing high resolution and detailed images. As the CMOS feature size decreases in new technologies, the size of the pixels shrinks or the spatial quality of the imaging improves for the

same pixel size.

Low power consumption. Advances in CMOS technology also provide for lower supply and threshold voltages. This allows the sensor to work with lower operational voltage, thus leading to a lower power consumption for imaging and processing circuits.

High integration capability. In addition to the sensor array, an image sensor requires timing and control circuits, signal processing units, analog-to-digital converter (ADC), and a digital interface circuit to extract, manipulate, and send the image data off chip. The CMOS technology provides for the development of all units on the same chip to make a camera-on-chip system. The potential of integrating all VLSI (very-large-scale integration) circuitry on a same chip decreases the packaging and component cost. Also, the on-chip or even in-pixel signal processing improves the overall speed of the imaging system.

Random access. To read the image data out of the camera, the pixel values should be read out. CMOS image sensors provide the possibility of the random access to the pixels for the read-out. Therefore, the image processing algorithms which might need only some of the pixels instead of the entire image data can read only the desired pixels to make more efficient implementations. Also, region of interest (ROI) imaging is easily implementable using control signals and on-chip control circuits.

On-chip signal processing and computational imaging. As mentioned before, in some applications, particular features of the image are required to be extracted instead of the entire high-quality image. By using CMOS technology for imaging, it is possible to combine imaging and all or some parts of image processing. By adding processing circuitry at the pixel, array, or chip level, a computational imaging system can be achieved which is more efficient compared to the imaging and off-chip signal processing. Advances in technology, improved performance of CMOS imagers, and also the consumer appetite for multi-mega

pixel and high-frame-rate cameras lead to the development of larger and faster CMOS imagers. However, these large-array imagers acquire huge amounts of imaging data that must then be processed. Therefore, an increasingly important signal processing task being developed and implemented in CMOS imagers is image compression. The large amount of image data and the limited communication bandwidth require image compression algorithms to decrease the overall throughput of the imager, which leads to lower power consumption and higher frame rate. The high integrability of CMOS imaging systems allows for the possibility of on-chip implementation of image compression algorithms. Therefore, in the next section, image compression and its different possible realizations for CMOS imaging are studied.

1.3 Image compression

Image compression algorithms use the spatial and temporal correlations in image and video signals to remove redundant information while keeping the essential features intact. Different algorithms such as Huffman, arithmetic, dictionary, differential, and transform coding as well as various quantization methods perform compression after image acquisition [21]. For computer implementation of compression algorithms, a digitized version of the image is used by the digital image processing units to convert high-resolution images to relatively small bit streams. However, the compression algorithms can also be implemented in hardware domain [22–24]. In this case, compression reduces the amount of the storage required for keeping or the bandwidth and power for transmitting of the data. But the throughput and frame rate of the camera is the same as the imager with no compression unless compression occurs inside the camera. Therefore, the compression can be performed on the same chip as the image sensor to improve the overall speed of the imaging system. Different

approaches can be used for the hardware implementation of on-chip image compression. These approaches are studied in this section.

1.3.1 On-chip digital implementation

This implementation could be performed in digital domain inside the camera chip [25, 26]. In this case, after the sensing and analog-to-digital conversion, the digital values should be stored and fed to the digital processing unit to perform the image compression. A schematic representation of the digital implementation is shown in Figure 1.6.

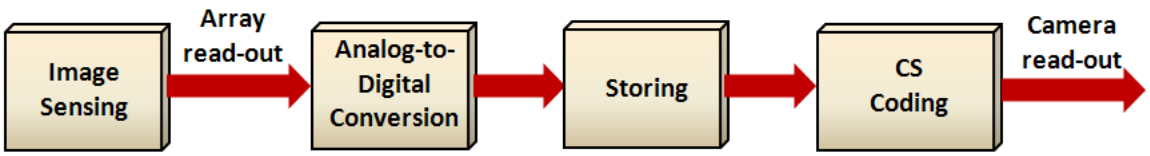


Figure 1.6: Block diagram for the digital implementation of image compression.

In this case, the VLSI implementation is performed on the same chip with the image sensor to reduce the amount of data which is sent off the chip. Although the camera read-out time decreases in this implementation, the array read-out time and amount of the storage are the same as an imager with no compression. The sensing and compression are sequentially performed. Therefore, there is a compression time period on top of the sensing time interval. Also, extra memory is required to keep the uncompressed image data and multiple memory access during the image compression leads to a high dissipated power. In addition, digital circuits use a considerable amount of the silicon area which can be used to increase the number of the pixels for the camera. Therefore, sequential implementation of sensing and digital compression is inefficient in terms of power consumption, time and silicon area.

1.3.2 Analog-level implementation

All or some parts of the compression could be performed in analog domain before or as a part of the analog-to-digital conversion. This structure decreases the amount of the on-chip storage area. Also, the analog implementations of the arithmetic circuits are faster and require less area and power compared to the digital circuits. Figure 1.7 shows the block diagram representation for the analog implementation.

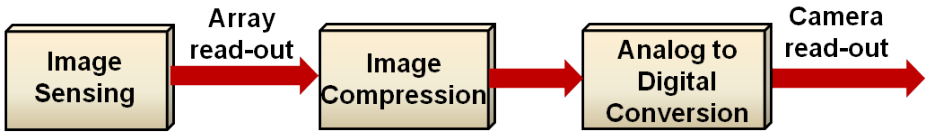


Figure 1.7: Block diagram for the analog implementation of image compression.

Several works have been presented on computational imaging and analog signal processing in image sensors [27–31]. These designs are intended to implement the image transformation and pixel manipulation in analog domain for general image processing tasks. By choosing an appropriate transformation, the chip can be used for image compression based on the transform coding approach [21]. The output of the transformation has a smaller number of significant coefficients compared to the number of pixels. Therefore, the digitized values of the significant coefficients are sent out of the imager instead of the pixels of the image as the compressed version of the image. In this case, the total number of analog-to-digital conversions are less as they are performed for the significant coefficients instead of the pixel values.

1.3.3 Compressive acquisition

In the digital and analog implementations mentioned in previous sections, the image processing is performed after the image acquisition phase. However, the photo sensing and image compression can be concurrently performed to reduce the analog or digital memories for storing the raw data and speeding up the entire imaging process [32–34]. Figure 1.8 shows the block diagram representation for this type of implementation termed *compressive acquisition*. To perform compressive acquisition implementation, all or some parts of the the analog computational circuitry could be placed inside the sensor array, i.e. focal plane processing. Also, the analog-to-digital conversion can also be performed inside the pixel by using digital pixels sensors (DPS). In this case, the output of the array is the compressed and digitized version of the image [35, 36].

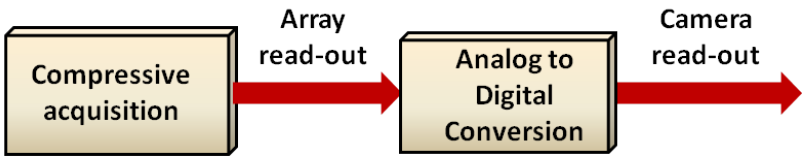


Figure 1.8: Block diagram for compressive acquisition implementation.

To implement more effective compression algorithms during sensing, more complex in-pixel circuits are required. There is a trade-off between the number of non-photosensitive elements inside the array and the time and memory which are saved. However, putting too many elements inside the pixels increases its size and leads to the lower resolution images. Also, adding extra components to the pixel decreases the ratio of the photosensitive area to the entire area of the pixel, i.e. the fill-factor. The lower is the fill-factor, the lower is the percentage of incoming light which is captured.

Therefore, compression methods with low complexity encoding process are of high interest for the realization of concurrent compression and sensing. The compressive sensing (CS) method addresses this problem. While other compressive acquisition methods try to implement conventional compression algorithms in parallel with sensing in circuit level, sensing the compressed version of the image is inherent in CS. Also, since the encoding part of CS method is straightforward, this leads to hardware-efficient implementation. The concept and different implementation of CS is discussed in the following sections.

1.4 Compressive Sensing

1.4.1 Background

The CS paradigm has attracted an increased interest in past decade because it intrinsically avoids sensing redundant information that exists in image or video data. Instead of sensing redundant information and removing them later during the compression step, only the required information is captured. In CS, a number of random projections of the image are sensed as the compressed version of the image, thus leading to a faster image acquisition system. Each projection, i.e. CS measurement, is a weighted sum of the level of the light for all pixels. The encoding process is only a matrix multiplication using random coefficients or the addition of some randomly selected pixels if binary random coefficients are used. Since the encoding process is straightforward, and the load of the computation is moved to the decoder, then the CS method is well-suited to be the preferred compression method where there are restrictions in power consumption, time and space on the encoder side.

CS-base imaging has also found extensive applications because of its impact in decreasing

the number of the sensors for a given image resolution and reducing the image acquisition time. Some of the main applications of the CS are in radar imaging [37, 38], DNA microarrays [39, 40], surface metrology [41], and biomedical imaging [42] including magnetic resonance imaging (MRI) [43–45], computed tomography (CT) imaging [46–48], ultrasound imaging [49–51], and fluorescent imaging [52].

As the CS paradigm involves the simultaneous sensing and compression, conventional architectures cannot be used for implementation of CS encoding. This is because in standard cameras, the level of the light for each spot in the scene, i.e. pixel value, is captured while the CS technique requires some transformed version of the light level of the entire scene. Therefore, new imaging structures have been proposed to implement CS-based imaging systems.

The first implementations of CS were in the optical domain. Here, the required transformation can be applied to the reflected light coming from the object. The resulting transform coefficients can be sensed by regular sensors and used in the decoder side to recover the pixel values for the image. High resolution imaging using low resolution sensor arrays and faster image acquisition are among the main advantages of the optical implementations. However, the implementation can be performed in electrical domain as well. Therefore, the CS method can be implemented in the focal plane of a CMOS sensor in the electrical domain. Although the electrical implementation does not lead to a lower number of sensors, the shorter acquisition time can be achieved with a smaller camera size compared to the optical-domain CS cameras.

Here, different hardware implementations of the CS-based image acquisition encoding in optical and electrical domains for image and video coding are discussed.

1.4.2 Optical implementation

Single-pixel cameras

Conventional cameras represent the captured image in spatial domain by using multiple photodetectors as the pixels of the image. Therefore, the resolution of the image increases with the number of photodetectors. However, the first implementations of CS imaging system used a single photodetector to capture different CS measurements from the scene. Optical devices were used to represent the pixels of the image. Therefore, the resolution of the image is determined in the optical domain.

A digital micro-mirror device (DMD) and CS were used to implement one of the first single-pixel cameras [53, 54]. Figure 1.9 shows the general schematic of the implemented single-pixel imaging systems. Each mirror in the DMD array is tilting towards or away from the photodetector based on an applied random coefficient [55]. The aggregate level of the light which is integrated by the photodetector represents one CS measurement, and different measurements will be achieved by different random alignments of the micro-mirrors. A multi-pixel image is recovered using all measurement values and the CS reconstruction methods. The idea of single-pixel imaging is also used in compressive confocal microscopes [56, 57].

In [58], a single detector terahertz imaging system is presented. A series of random masks have been exploited to make the CS measurements. Each mask represents a random pattern using a combination of transparent and opaque areas corresponding to random coefficients. In [59], structured light and a single detector were used to measure specific features of the image. Instead of using uniform illumination for imaging, a set of spatially structured illumination patterns have been used to extract a sequence of measurements from the image.

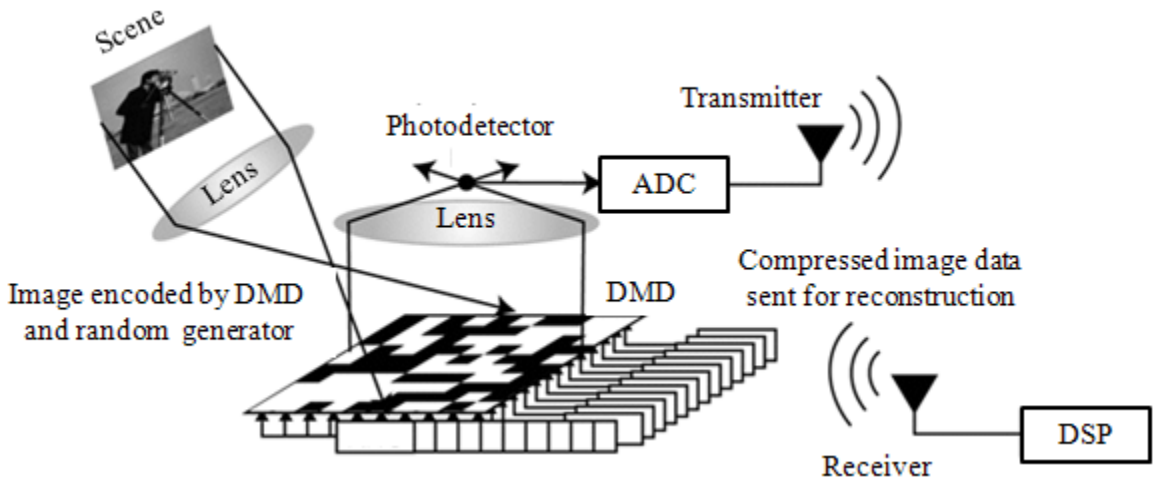


Figure 1.9: General schematic of single-pixel camera set-up (adopted from [54]).

All the single-pixel cameras described above are based on serial measurement extraction. Different measurements are made during consecutive time intervals. Therefore, the object should be fixed during the measurement process. This stationary requirement makes the designs unsuitable for video acquisition when there is high level of movement in the scene from one measurement to another.

Coded aperture cameras

Typical cameras use pinhole apertures and lenses to focus the incoming light from the object on to the sensor array. Each sensor in the array represents a pixel of the image for the object. However, the optical field from the scene could be modulated using a coded structure for the aperture. Therefore, some specific image transformations can be implemented in optical domain [60]. Considering the CS measurements set as a random transformation

of the image, then coded aperture can also be used in CS cameras. In [61], a high resolution coded aperture with a low resolution focal plane array aperture has been used for imaging. The aperture has been designed so that different CS measurement values are collected by the different pixels of the low resolution array. Therefore, the value measured by each sensor on the focal plane array is a CS measurement value formed in optical domain by coded aperture. In contrast to the single-pixel imager, all CS measurement values are extracted in the same time interval.

The coded aperture approach is also used for spectral imaging. In [62], a single-shot spectral imager using the CS framework was presented. To improve the signal-to-noise ratio (SNR) of reconstruction, a multi-shot approach with multiple apertures and focal plane measurement sets is presented in [63]. The CS measurement can also be implemented by using a random mask placed on the lens. Here, the convolution of the image and mask signal is made at a single exposure to create the CS measurement values [64]. Figure 1.10 shows the spectral imaging structure, [62], and random mask set-up presented in [64].

Random lens imaging

An imaging system developed in [65] utilized random lens made from multi-faceted mirrors for compressive image acquisition. The system is the same as in conventional cameras, but the input-output relationship of the light rays is randomized. The only extra hardware is some small mirror patches stitched around the sensors, thus leading to an ultra-thin design for multi-spectral and high dynamic range imaging [65].

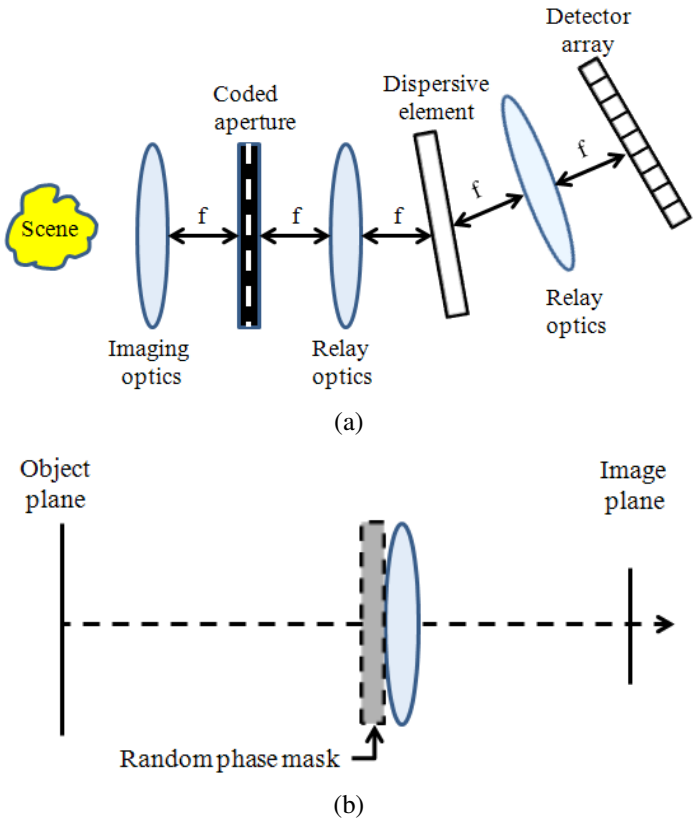


Figure 1.10: (a) Schematic of the single dispersive spectral imaging system [62], and (b) Optical set-up for random-mask image acquisition [64].

1.4.3 CS-CMOS cameras

In the hardware implementation of CS encoding discussed above, the random transformation required for CS measurements has been performed in optical domain. This leads to an optimum use of photodetectors to achieve high resolution imaging using low resolution sensor arrays at the cost of the larger camera size due to the optics. However, on-chip implementation of the CS algorithm in electrical domain is more useful for portable applications where there are limitations on the size of the camera and its power consumption. The CS implementation could be based on the block diagrams shown in Figures 1.6 and

1.7. In this case, the random selection and summation of the pixels can be performed after light integration [66, 67].

In [66], a block-based CS encoding for digital pixel sensors was presented. The sensor array was divided into different blocks and one pixel is randomly selected in each block as the CS measurement for that block. The selected digital outputs were then saved in 8-bit memories to be used in the CS image reconstruction process.

Another block-based implementation of CS at the ADC level was presented in [67]. The random measurement was performed by using a CS multiplexer before a $\Delta\Sigma$-based ADC. Row and column block selectors were used to connect different blocks to a CS multiplexer and a pseudo-random generator provided the random coefficients required for the CS multiplexer. Although the measurement process was not performed in parallel with sensing, different measurements were extracted in parallel. This led to a single-shot imaging scheme.

The main advantage of the CS encoding, the simultaneous sensing and compression, is disregarded in both above structures in [66] and [67]. To be compatible with the CS concept, the compression should be performed at array level and before the array read-out, Figure 1.8. The random sequence should be applied at the beginning of the integration time to make sure that the corresponding CS measurement was integrated during the light integration. Therefore, one of the major difficulties in realizing the CS-CMOS implementation shown in Figure 1.8 was the method of applying random coefficients to the pixels. It is because the random bits change for different measurements and should be individually applied to the pixels. Different techniques of random selection- in-pixel random generator, column-row random selection, and block random selection, are now discussed.

In-pixel random generators

Some designs use the in-pixel memories or digital circuits to put all or part of the random generator inside the pixel. However, by putting non-photosensitive elements inside the pixel, the fill-factor and sensitivity of the imager decrease. Also, the size of the pixel increases, thus leading to the lower image spatial resolution.

In [68], in-pixel memories with control logic, a finite state machine (FSM), 3 LFSR's (linear feedback shift register) and a single analog processing unit were used to perform in-pixel convolution in real-time. Each pixel contained a flip flop as the local memory plus control logic to implement the horizontal and vertical shifting. Using a two-dimensional scrambling technique, the random coefficients were applied in two dimensions over the entire array. Also, the measurement process was performed in two dimensions using the differential current output of each pixel which were connected to the corresponding outputs from other pixels. Finally, the accumulated differential currents were fed to a transimpedance amplifier (TIA) to calculate the final CS measurement. The measurement process is completely performed during the light integration in analog domain, although the scrambling technique needed extra time between the acquisition of one measurement and the next.

Column-row random selection

Random coefficients can be fed to the pixels from outside the array. However, because of layout restrictions, having individual access to each pixel is impractical for a large array sensor. This is due to the limited number of the metal layers in current CMOS technologies and the fact that the metal layers cannot pass through the photosensitive areas. Therefore, different sub-regions of the pixels should share common paths to access the coefficients.

Random coefficients can be partially fed to the array along the columns so that pixels in each row have the same coefficient. Then, the outputs of the columns can be randomly combined outside the array to complete the random measurement.

In [69, 70], a separable-transform image sensor which is also capable of implementing the CS encoding was designed. Instead of sensing the pixel values for the image, the imager projects the image on a specific basis and produces the projection coefficients. The convolution and image transformation were performed for separate sub-regions of the image with 8×8 and 16×16 block sizes, respectively. The main feature of the design was to exploit the separability property to perform image transformation in two steps: 1) focal plane processing along the columns and during the sensing, and 2) analog computational units outside the array and before the ADC converter.

The image transformation was divided into two separate matrix multiplications. The first computation was performed by using differential transistors in the pixels and using Kirchhoff's current law along the columns. The coefficients for each pixel were provided by the differential signal paths which were the same for the pixels in each row. The results for different columns were fed to an analog vector-matrix multiplier (VMM) to perform the second part of the matrix computation. The imager can be exploited for CS imaging by dividing the random multiplication of CS encoding into two separate matrix computations. However, the separability property and analog multiplication was too complicated for the CS implementation, although it is useful in performing other image transformations.

Switched capacitor circuits were used in [71] to implement a separable transform imager. By adding one more transistor and one capacitor to the pixel structure and using switched capacitor circuit for each column, the random combination of the pixels for each column was achieved. The calculated values from all columns are fed to another switch capacitor

circuit to make the final random measurement. The timing of the switching clocks can be adjusted to choose one coefficient from the set $0, +1, -1$ for each pixel.

An implementation of CS encoding using random convolution was presented in [72]. The structure is a combination of in-pixel and column-row random selection. Binary random coefficients were provided by in-pixel memories and a LFSR to feed the initial values for the in-pixel memories. Each pixel contained a photodiode and a 1-bit memory which was connected to the memories in adjacent pixels to make an embedded shift register inside the array. The outputs of the pixels in each column were connected together, and Kirchhoff's current law applied along the columns to make a random light integration for each column. A time-domain multiplexer and an ADC were used to digitize the outputs for different columns. The rest of the measurement process was performed by doing several digital summations. To complete the randomness of the convolution process, a pseudo-random triggering LFSR was used after column integration, but before column read-out. Note that the CS measurement process was performed in two steps, with the second step in digital domain and after the light integration. This is not the simultaneous computation and sensing expected in CS encoding.

Block random selection

To feed the random coefficients from outside the array, the metal paths could be shared among different blocks of the images instead of the columns. As the size of the block was small compared to the entire array, individual access to all pixels in the block from outside the array was feasible. Also, each pixel in the block could share the metal path with its corresponding pixels in other blocks. Therefore, each random coefficient was connected to the all corresponding pixels in different blocks. It is analogous to the column-row structure

when all pixels in one column share a same random coefficient. Therefore, the array can be divided into separate blocks and the CS encoding can be implemented for different blocks separately. Individual access to all pixels in each block can be possible at the measurement time of the block [73, 74].

Table 1.1 shows a summary of different CS-CMOS imager architectures discussed above. PPS (passive pixel sensor), APS (active pixel sensor), and DPS (digital pixel sensor) were used for different implementation. For PPS sensors, the charge-to-voltage conversion is performed outside the pixel while it is inside the pixels for APS sensors. This leads to the higher noise performance for APS in the cost of lower fill-factor. For DPS pixels, the analog-to-digital conversion is performed inside the pixel. DPS has the lowest fill-factor compared with the two other structures, but the imaging systems is faster [75].

The implementations can be in analog or digital domain. In the digital implementation, the random selection and measurement extraction were straightforward as they used the digital data outside the array. However, analog implementation led to a faster and more efficient encoding process. As can be seen in the table, different works used different methods of random selections. In-pixel random selection led to the use of extra digital components inside the pixel which reduced the fill-factor of the design. However, column-row and block random selection methods used the external random coefficients to avoid internal digital circuits. Also, the block method is more efficient in terms of in-pixel and overall hardware usage.

Table 1.1: Summary of various CS-CMOS imagers available in the literature.

Ref.	**Pixel Type**	**Tech. (CMOS)**	**Array Size**	**In-pixel Components**	**Technique**
[66]	DPS	–	–	1 Transistor, 2 Invertors, 1 And gate, 1 Comparator	Digital domain, Random selection after integration
[67]	APS	0.15 μm	256×256	4 Transistors	A/D level, Random selection after integration
[68]	APS	0.18 μm	256×256	3 Transistors, 3 Nand gates, 1 D flip-flop	Analog domain, In-pixel random selection, Differential current & TIA
[69]	PPS	0.35 μm	256×256	2 Transistors	Analog domain, Column-row random selection, Differential current & VMM
[71]	APS	0.5 μm	128×128	4 Transistors, 1 Capacitor	Analog domain, Column-row random selection, SC circuits
[72]	PPS	–	–	1 Transistor, 1 Flip-flop	Analog & digital domain, In-pixel & column-row random selection

1.5 Outline of thesis

The rest of the thesis is arranged as follows. In chapter 2, the image recovery problem of CS method is studied. The mathematical model and different methods of CS reconstruction are investigated and then a modified optimization method for the reconstruction is presented. A current-mode implementation of the CS using the idea of APS with integrator is presented in chapter 3. In Chapter 4, the design and implementation of a voltage-mode CS-CMOS image sensor is described. In both chapters 3 and 4, the block-based implementation are used, and the designs are fabricated in 130nm CMOS technology. Also, simulation and experimental results are presented to evaluate the validity of proposed CS-CMOS imagers. In chapter 5, the CS encoding methods for video coding is investigated. The chapter starts with an overview on some works on CS video coding, especially the high-frame rate multiple-camera CS video coding method. Then, the pixel and array structure which are compatible with the mentioned video coding method are presented. The conclusion of the thesis with comments and recommendations on future research are presented in chapter 6.

1.6 Contributions

The research presented in the thesis has led to several publications. Here are some highlights of the contributions. This thesis uses some of the material presented in the publications mentioned below.

- In last decade, compressive sensing (CS) was considered as an efficient way of image

acquisition. However, conventional imaging systems cannot be used for the implementation of CS imaging. Therefore, several approaches in optical and electrical domain were used to realize hardware implementation of the CS. In [76], we have presented an extensive review on different implementations of CS method. Considering the recent advances in CMOS technologies and the feasibility of performing on-chip signal processing, practical issues in the implementation of CS for CMOS sensors has been emphasized.

- The capabilities of the image reconstruction method is of high importance in a CS imaging system. By using high performance algorithms, images with higher quality can be recovered for the same compression ratio. Therefore, an extensive research is in progress to improve the performance of CS reconstruction algorithms. In [77], we modified the conventional CS reconstruction methods to improve the quality of the decoded image.

- The previous CS-CMOS imagers presented in literature used extra in-pixel elements or performed all or some parts of the CS encoding after the acquisition, leading to a non-real-time implementation. We presented a fast and on-the-fly current-mode implementation of CS with minimum number of in-pixel elements in [74]. The block-based implementation provided the possibility of moving the measurement circuitry outside the array to make the design area-efficient. Using 130nm technology, the design was fabricated and measured.

- Another block-based, on-the-fly implementation of CS-CMOS imager using voltage-mode structure is presented in [73]. This implementation provides superior noise

performance compared to our other design, but the cost was one more in-pixel transistor.

- All previous CS video encoders implemented the CS coding for different frames of the video separately. The CS measurements were extracted in spatial domain for each frame as a separate image. In [76, 78], we presented a new CS measurement hardware for making the CS measurements in spatial and temporal domain. This leads to a high-frame-rate video capture system.

Chapter 2

Compressive sensing image reconstruction

2.1 Compressive sensing encoding

In the compressive sensing paradigm, instead of capturing the image data for every pixel and then using the intensity values of the pixels for image compression, a number of combinations of those values, i.e., CS measurements, are captured as the compressed version of the image. The output of the imager is then a sequence of CS measurements in which each measurement represents a random combination of the light intensity over the entire array. To study the reconstruction problem of CS-encoded images, there should be a mathematical representation for the encoding problem.

Considering each block of the image as the vector $\mathbf{x}$, which is formed by column concatenation for notational convenience, and by defining $\mathbf{\Phi}_{M\times N}$ as the measurement matrix, the compression process can be expressed by the following linear equation:

$$\mathbf{y} = \mathbf{\Phi x}, \tag{2.1}$$

where **y**, the measurement vector, is the output of the encoder. Also, M and N are the number of measurements and length of the vector **x**, respectively. Compression occurs when the number of measurements is less than the number of pixels, i.e., $M < N$.

As can be seen in (2.1), the entire encoding process is a simple matrix multiplication. Although we assumed that all values for the pixels- elements of vector **x**, are available to perform the encoding, we will see in the next chapters that the measurement values can be directly obtained from the image sensor without having the pixel values in advance.

The measurement matrix, $\mathbf{\Phi}$, should be independent of the signal **x**. Since $M < N$, the problem of recovering the image using its corresponding measurements and the measurement matrix appears to be ill-conditioned. However, as described in [79], the measurement matrix should be a random matrix to make the problem well-conditioned. As an example, the elements of $\mathbf{\Phi}$ could be independent, identically distributed random variables from a Gaussian distribution or binary values from a uniformly distributed set of random variables.

In the case of using binary coefficients, each row of the measurement matrix $\mathbf{\Phi}$ is a binary random sequence and its corresponding measurement value is a summation of the values of some randomly chosen pixels, i.e., pixels corresponding to the 1's in the corresponding row of $\mathbf{\Phi}$. Therefore, there is no multiplication and the measurement process is simply the addition of some randomly selected pixels. Each measurement can be expressed by the following equation.

$$y_i = \sum_{j|\phi_{ij}=1} x_j, \tag{2.2}$$

where y_i is the i^{th} measurement, ϕ_{ij} is the j^{th} element of a vector of independent, identically distributed variables, and x_j is j^{th} element of vector **x**.

2.2 Compressive sensing decoding

2.2.1 Basic pursuit methods

Natural images consist of highly structured information or a strong dependency between the adjacent pixels in the image. This dependency can be mathematically investigated by evaluating the expansion coefficients of the image in a special basis, $\mathbf{\Psi}$ (e.g., discrete cosine transform (DCT), discrete wavelet transform (DWT), contourlet transform.) Considering an N-pixel image as the vector $\mathbf{x}$, then the transformation of the image in basis $\mathbf{\Psi}$ is:

$$\Theta = \mathbf{\Psi x}. \tag{2.3}$$

In (2.3), $\Theta = (\theta_1, \theta_1, ..., \theta_N)$ is the vector of expansion coefficients. The image $\mathbf{x}$ is called k-sparse in $\mathbf{\Psi}$ domain when only k expansion coefficients are significant and the other *(N-k)* coefficients are zero or negligibly small. Therefore, although the image is represented by N pixel values in the grey level domain, it could be represented by k values in $\mathbf{\Psi}$ domain. Conventional transform coding schemes apply the transform $\mathbf{\Psi}$ to the image and keep the k significant coefficients as the compressed version of the image. The image can be recovered in the decoding process using these coefficients and the inverse transform $\mathbf{\Psi^{-1}}$. However, all N pixels should be acquired to extract the significant coefficients. It is not efficient to capture N values while only k values are enough for the image reconstruction. Compressive sensing addresses the problem of inefficient sampling and then compression.

The important part is that the measurement matrix $\mathbf{\Phi}$ should be properly chosen to extract the maximum information from the image. The measurement matrix can be chosen based on the particular properties of the image. However, the structure of the image is unknown

unless the CS is used for the compression of an available image, but this is not the main purpose of CS. Also, the structural information of the image might change from one image to another. Therefore, a universal measurement matrix which is applicable for all images is required.

It has been shown that when the number of the measurements, M, is greater than the order of sparsity of the image, k, the necessary and sufficient condition for the image reconstruction is the validity of restricted isometry property (RIP) for the measurement matrix [79–82]. The RIP property is expressed as follows.

$$1-\delta < \frac{||\mathbf{\Phi}\mathbf{\Psi}^{-1}\mathbf{v}||_2}{||\mathbf{v}||_2} < 1+\delta, \tag{2.4}$$

where $\mathbf{v}$ could be any vector with the same non-zero coefficients as Θ, and $0 < \delta < 1$. It can be shown [82] that when $\delta < \sqrt{2}-1$, there is an upper bound on the reconstruction error in the image recovery problem. This means that the image can be reconstructed.

The measurement matrix $\mathbf{\Phi}$ should also be incoherent with the basis $\mathbf{\Psi}$ [79]. These two properties will be satisfied when the entries of the measurement matrix are independent and identically distributed random variables. Therefore, the CS encoding process can be expressed as the extraction of a set of weighted addition with random weights.

Reconstruction algorithms exploit the measurement vector, $\mathbf{y}$, measurement matrix, $\mathbf{\Phi}$, and the structural information of the image to recover the value of each pixel in the image. The reconstruction algorithm searches for the sparsest solution that matches the measurement values. The 0-norm (ℓ_0) of the transformed version of the image is the best criterion to measure the level of the sparsity. Therefore, the problem can mathematically be expressed as follows.

$$Minimize_{\mathbf{x}}||\mathbf{\Psi}\mathbf{x}||_0 \quad subject\ to \quad \mathbf{y} = \mathbf{\Phi}\mathbf{x}. \tag{2.5}$$

However, ℓ_0 minimization is an NP-hard problem. Therefore, a large number of studies on CS theory have focused on the design of CS reconstruction methods with appropriate computational complexity and acceptable reconstruction performance. It can be shown that instead of the ℓ_0 minimization, the 1-norm (ℓ_1) minimization and linear programming can be exploited for CS reconstruction if the measurement matrix satisfies the RIP.
Therefore, considering the addition of the noise on the measurements values, the reconstruction problem can be expressed as,

$$Minimize_{\mathbf{x}} ||\mathbf{\Psi x}||_1 \quad subject\ to \quad ||\mathbf{y} - \mathbf{\Phi x}||_2 < \epsilon. \tag{2.6}$$

In (2.6) ϵ is the upper bound on the noise magnitude. Even with the noisy measurements, the image could be reconstructed and the reconstruction error is bounded by the noise level in addition to the deviation from sparsity [82]. To solve the convex optimization problem [83] in (2.6), basic pursuit (BP) methods [84] that use the linear programming approach can be employed.

2.2.2 Total variation minimization

One of the inherent properties of typical images is their small total variation (TV) which can be defined as 1-norm of discretized gradients as follows [85]:

$$\begin{aligned} \mathbf{TV} &= \textstyle\sum_{i,j} ||(\nabla \mathbf{I})_{ij}||_1 = \sum_{i,j} (|I(i,j) - I(i-1,j)| \\ &+ |I(i,j) - I(i,j-1)| + |I(i,j) - I(i+1,j)| \\ &+ |I(i,j) - I(i,j+1)|). \end{aligned} \tag{2.7}$$

Among all vectors which satisfy the measurement equality, the TV minimization algorithm selects the vector with smallest total variation [85]:

$$Minimize_{\mathbf{x}} \|\mathbf{Tx}\|_1 \quad subject\ to \quad \mathbf{y} = \mathbf{\Phi x} \tag{2.8}$$

where $\mathbf{T}$ is a $4N \times N$ matrix to calculate total variation so that $\mathbf{T}(i, j) \in \{-1, 0, 1\}$. This problem is also a 1-norm minimization with a different coefficient matrix, $\mathbf{T}$, instead of $\mathbf{\Psi}$.

2.2.3 Second-order cone programming

Any convex optimization problem can be described by some standard forms. The reconstruction problem of CS can be expressed as the standard second-order cone programming, (SOCP), convex optimization problem as follows [83]:

$$\begin{aligned} Minimize_{\mathbf{x}} \quad & \mathbf{C}^T\mathbf{x} \\ subject\ to \quad & \|\mathbf{A}_i\mathbf{x} + \mathbf{b}_i\|_2 \leq \mathbf{e}_i{}^T\mathbf{x} + \mathbf{d}_i, \quad i = 1, ..., L, \end{aligned} \tag{2.9}$$

Where, vector $\mathbf{x}$ is the optimization variable. $\mathbf{C}$ is the coefficient matrix for the objective function, and $\mathbf{A}_i$, $\mathbf{b}_i$, $\mathbf{e}_i$, and $\mathbf{d}_i$ are the coefficients for the i^{th} constraint function. Note that adding extra linear constraints keeps the problem as a SOCP one.

Considering the optimization variables $\mathbf{t}$ and $\mathbf{x}$, and by accepting a small error in the equality constraint, the problems of TV and 1-norm minimization could be rephrased as a SOCP problem:

$$\begin{aligned} Minimize_{\mathbf{x},\mathbf{t}} \quad & \mathbf{1}^T\mathbf{t} \\ subject\ to \quad & \text{-}\mathbf{t} \leq \mathbf{Mx} \leq \mathbf{t} \\ & \|\mathbf{y} - \mathbf{\Phi x}\|_2 \leq \epsilon, \end{aligned} \tag{2.10}$$

where $\mathbf{M}$ equals $\mathbf{T}$ or $\mathbf{\Psi}$, ϵ is a sufficiently small value and $\preceq$ is componentwise inequality.

2.2.4 Greedy methods

Linear programming leads to promising results in terms of reconstructed image quality. However, it is of high computational complexity, for instance in order of $O(M^2N^{\frac{3}{2}})$ [86]. Some other fast convex programming methods, for instance gradient methods [87], have been presented to improve the computational load of the reconstruction problem. Also, some efficient iterative methods based on solving several optimization sub-problems with sparsity regularizer have been proposed in [88, 89] to improve the convergence time of the optimization problem.

However, there is another family of algorithms based on iterative greedy methods which are computationally efficient and easy to implement. However, the cost is lower image quality for the same number of measurements [90–95]. Greedy methods are based on an iterative, non-optimization approach by successive approximation of the data and its residuals. Orthogonal matching pursuit (OMP) [90], stagewise OMP (StOMP) [91], and regularized OMP (ROMP) [92] are some greedy methods with computational complexity of order O(kMN) [93]. Subspace pursuit (SP) [93], compressive sampling matching pursuit (CoSaMP) [94], and adaptive sparsity matching pursuit (ASMP) [95] are some other greedy methods with comparable reconstruction quality to BP methods.

Multi-core processors, shared memories and parallel processing systems can also be used to improve the execution time of the reconstruction problem [96]. Also, in applications where there are limitations in power consumption and capabilities of local processors, e.g. smart phones, the reconstruction step can be performed on a remote cloud-computing server [97].

2.3 Modified optimization problems

The TV minimization method exploits the sparsity of the gradient of the image to find the optimum point in its optimization problem. Therefore, this method disregards the sparsity and energy compaction of the image in other domains like DCT or DWT. Also, 1-norm minimization exploits just one kind of sparsity in its cost function.

To find a more image-like vector as the optimum point, different kinds of image properties can be considered in the same problem. Intuitively, the problem is looking for a vector in which its transformation has the smallest 1-norm among all vectors specified by all constraints of the problem. The measurement constraint specifies a set of vectors which are close to the original one in terms of the measurement matrix. Some vectors belonging to this set other than the vector corresponding to the original image, might minimize the cost function even though they are not appropriately image-like vectors. Some of these vectors could be removed from the feasible set of the minimization problem by using some extra constraints related to other properties of natural images. Finally, by finding the best compromise among different constraints of the problem, the performance of the reconstruction can be improved.

To keep the problem as the standard SOCP one, several 2-norm constraints may be added to TV or 1-norm minimization problems to include the effect of the energy compaction of image in other domains. This removes some irrelevant vectors from the feasible set. Thus, the modified minimization problems with new constraints and the same cost function are expected to improve the performance of the reconstruction. The optimization problem with

new constraints is expressed as follows.

$$\begin{aligned} Minimize_{\mathbf{x},\mathbf{t}} \quad & \mathbf{1}^T\mathbf{t} \\ subject\ to \quad & -\mathbf{t} \leq \mathbf{M}\mathbf{x} \leq \mathbf{t} \\ & \|\mathbf{y} - \mathbf{\Phi}\mathbf{x}\|_2 \leq \epsilon \\ & \|\mathbf{F}_i(\mathbf{x})\|_2 \leq T_i, \quad i = 1, ..., c. \end{aligned} \tag{2.11}$$

where c is the number of new constraints, $\mathbf{F}_i(\mathbf{x})$ is a linear function of $\mathbf{x}$ and T_i is the threshold corresponding to the i^{th} constraint.

2.3.1 Discrete cosine transform (DCT) constraint

One of the most common properties of natural images is the energy compaction of DCT coefficients of the image. Absolute values of the DCT coefficients decrease by moving from the top to the bottom and from the left to the right of the DCT matrix. Therefore, the right and bottom half, or right bottom quarter, of the DCT matrix contains many small elements. Therefore, regarding the sparsity of DCT matrix, the following constraint can be added to 1-norm and TV minimization method.

$$\|\mathbf{A}_{rb}DCT(\mathbf{x})\|_2 \leq T_{DCT} \tag{2.12}$$

where $\mathbf{A}_{rb}$ is a matrix to yield right bottom quarter of the DCT coefficients matrix by using its vectorized version, $DCT(\mathbf{x})$.

2.3.2 Discrete wavelet transform (DWT) constraint

Wavelet transform is being used in a number of different applications such as digital filtering and image compression. The energy compaction property of DWT makes it suitable for compression algorithms. Considering DWT as a combination of consecutive low and high-pass filters, and for natural images, most of the energy is compacted into lower bands. Thus, the outputs of consecutive high-pass filters are sparse. This sparsity can be expressed as a new constraint as follows.

$$||DWT_{HH}(\mathbf{x})||_2 \leq T_{DWT} \tag{2.13}$$

where $DWT_{HH}(\mathbf{x})$ is the output of consecutive high-pass filters.

2.3.3 Contourlet constraint

Contourlet transform is a combination of Laplacian Pyramid (LP) and directional filters [98]. As the fine output of LP is fed to the directional filter bank, the output of different filters are the high frequency components of the image in different directions. High frequency bands of natural images contain less energy in most of the directions. Therefore, the following sparsity constraints can be considered for the 1-norm and TV minimization algorithms.

$$\frac{4}{N}||\mathbf{D}_{Fi}\mathbf{L}_d\mathbf{x}||_2 \leq T_{i,CL} \quad i = 1, ..., \ell. \tag{2.14}$$

where $\mathbf{L}_d$ provides the difference signal in LP and $\mathbf{D}_{Fi}$ provides the output of the i^{th} directional filter.

2.4 Progressive Thresholding

There are thresholds in all the new constraints proposed in the previous sections. The values of these thresholds have a significant role in the feasibility of the problem and amount of improvement caused by the addition of the new constraints. Geometrically, the 1-norm function in the CS reconstruction problem is a combination of hyperplanes around the origin which are defined by the coefficient matrix, i.e., $\mathbf{T}$ or $\mathbf{\Psi}$. Measurement and additional constraints are ellipsoids centered on the measurement vector, $\mathbf{y}$, and the origin, respectively. Therefore, the minimization problem is basically about finding a point on the closest hyperplanes to the origin that lies on the intersection of ellipsoids, i.e., the feasible set.

Considering $T_{DCT} = \infty,\ T_{DWT} = \infty,\ or\ T_{i,CL} = \infty$, then the new constraints do not have any effect on the performance of the reconstruction and the problem becomes one without the new constraints. In fact, the measurement ellipsoid completely lies inside the other ellipsoids, thus making the additional constraints ineffective. On the other hand, for $T_{DCT} = 0,\ T_{DWT} = 0,\ or\ T_{i,CL} = 0$, the problem is infeasible because there is no intersection among the constraint sets. Any threshold between the two extremes mentioned, i.e., $0\ and\ \infty$, should be appropriately determined to yield a non-empty intersection while increasing the performance of the reconstruction.

In addition to the above mentioned thresholds, threshold of the measurement constraint, ϵ, has a considerable effect on the feasibility and performance of the problem. A smaller ϵ emphasizes more on the validity of the measurement constraint while a higher value relaxes this constraint. Considering the reconstruction problem with only the measurement constraint, there is an $\epsilon = \epsilon_0$ for which the best reconstruction is achieved.

To find the threshold for the new constraints, it is assumed that there is just one new constraint or equivalently, all the thresholds are the same. Figure 2.1 illustrates the flowchart of

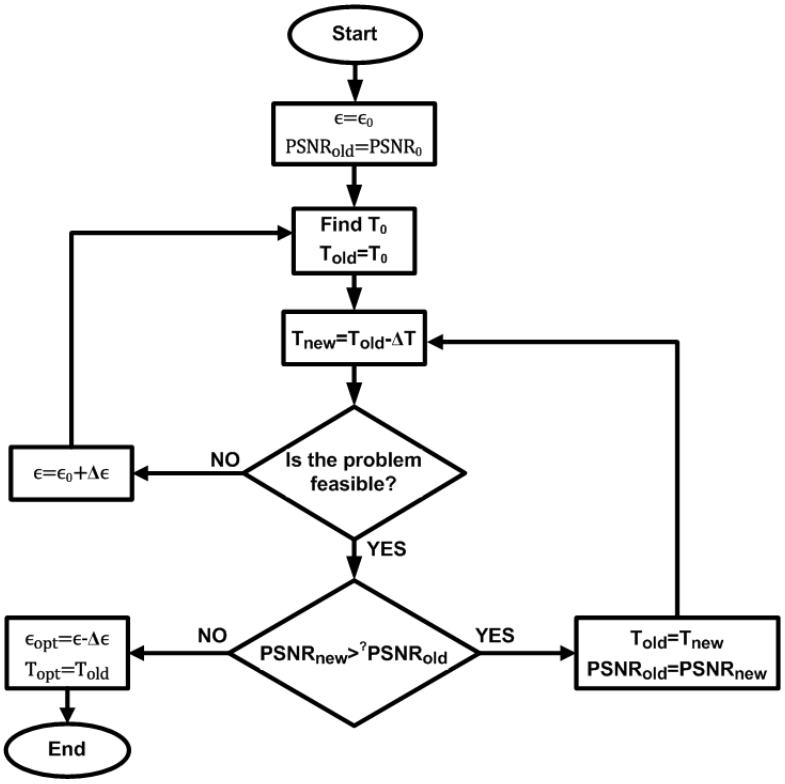


Figure 2.1: Progressive thresholding.

this progressive algorithm to find the optimum threshold. In the first step, a high threshold, T_0, is selected so that the problem works as a regular minimization algorithm with only the measurement constraint. In the next steps, the threshold is reduced by ΔT to the extent that there is no more improvement on reducing the threshold, or the problem becomes infeasible. To evaluate the performance of reconstruction in each iteration, $PSNR$ is defined as

follows based on the reconstructed vector, $\hat{\mathbf{x}}$, and the measurement vector, $\mathbf{y}$.

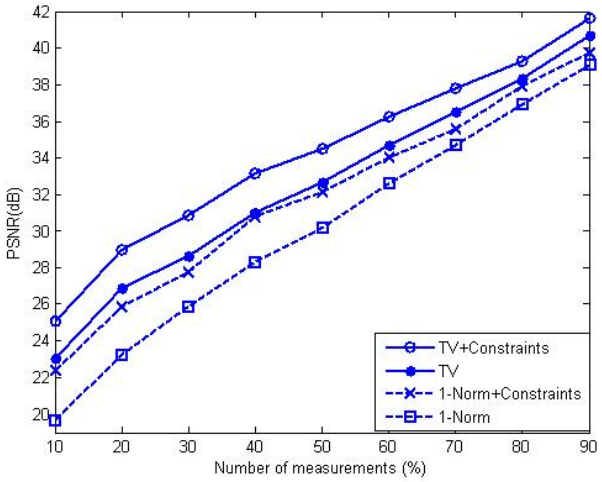
$$PSNR(\mathbf{y}) = \log M \frac{max(\mathbf{y})}{\|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2} \tag{2.15}$$

When the problem is still feasible and the highest $PSNR$ is attained, $PSNR_{old} > PSNR_{new}$, the corresponding threshold is the optimum value. This threshold makes the best compromise between measurement and the additional constraints. If the highest $PSNR$ is not reached and the problem becomes infeasible, then the measurement constraint can be relaxed more. This is achieved by increasing ϵ by $\Delta\epsilon$ and a new T_0 is found for the new ϵ. While keeping the amount of $PSNR_{old}$, the algorithm is repeated to find the optimum thresholds for all constraints in the problem, i.e., ϵ and T.

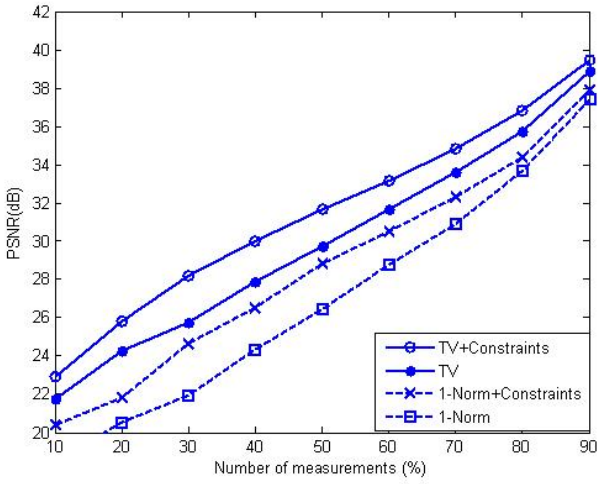
2.5 Simulation results

In this section, the performance of different reconstruction problems expressed by (2.11) is investigated. Considering energy compaction of *DCT*, *DWT* or contourlet transform of the image as a new constraint, i.e., $\mathbf{F}_i(\mathbf{x})$, and 1-norm (with $\mathbf{\Psi}$ set to *DCT*) or total variation of the image as the cost function, 6 different reconstruction methods are evaluated.

For the compression process, an $M \times N$ binary matrix of random elements has been exploited. Each element is made by rounding a Gaussian distributed value between 0 and 1. Also, CVX , a MATLAB software for convex programming, was used to solve the SOCP problem for each CS reconstruction. The wavelet transform with 9-7 biorthogonal filters [99] and two decomposition levels were used. For the contourlet transform, 9-7 biorthogonal filters for the LP stage and four directional filters, i.e., $\ell = 4$, were exploited. Figure 2.2 shows the the graphs of $PSNR$ versus the number of measurements for four different CS



(a) Lena



(b) Fishing boat

Figure 2.2: PSNR vs. the number of CS measurements (%)

recovery methods. In each case, among all three possible constraints, i.e., DCT, DWT or counterlet, the one with the best reconstruction quality was considered. By comparing the results for the different methods, it can be found that adding new constraints to either the TV or 1-norm method consistently improves the performance of reconstruction. Also, the

TV method with new constraints are superior to other methods.

Figures 2.3 and 2.4 show the CS recovered images for the Lena and Fishing boat images respectively. Again, the TV method with new constraints leads to the best performance. Note that the visual improvement is more clear at the edges of the images compared to regular TV minimization method.

As mentioned in the previous sections, the reconstruction performance of the TV method is usually better than 1-norm, but for images with large areas with high frequency components, the performance of the TV method degrades. Figure 2.5 shows the CS recovered image for the Barbara image. The quality of the 1-norm recovered image is better than the TV one, especially for the cloth in the image. Although adding new constraints to the TV problem improves its performance, the 1-norm with new constraints outperforms the other three methods.

To quantify the amount of improvement caused by the different types of new constraints, Table I shows the PSNR of reconstructed image, equation (2.16), for different methods and three images, i.e., Lena, Barbara and fishing boats.

$$PSNR = \log N \frac{max(\mathbf{x})}{||\mathbf{x} - \hat{\mathbf{x}}||_2} \tag{2.16}$$

As interpreted from the table, adding new constraints results in a improvement, up to 2.5dB in the 1-norm and TV minimization algorithms. These improvements demonstrate to the effectiveness of the new constraints explained in the previous sections. As shown in the table, the contourlet constraint gives the best performance among all the methods discussed because of its effectiveness in capturing smooth contours and the geometrical structure of the image.

Table 2.1: PSNR (dB) of reconstruction (40% measurements)

Images	**New Constraint**	No constraint	DCT	DWT	Contourlet
Lena	**1-Norm**	28.33	30.397	30.42	30.77
	TV	30.996	32.641	32.762	33.155
Barbara	**1-Norm**	26.743	27.99	28.02	28.44
	TV	24.914	26.214	26.331	26.816
Fishing boat	**1-Norm**	24.342	26.012	26.254	26.559
	TV	27.896	29.523	29.721	29.996

(a) Original

(b) 1-Norm (28.33dB)

(c) 1-Norm+new constraint (30.77dB)

(d) TV (30.996dB)

(e) TV+new constraint (33.155dB)

Figure 2.3: CS recovered Lena image (40% measurements)

(a) Original

(b) 1-Norm (21.97dB)

(c) 1-Norm+new constraint (24.637dB)

(d) TV (25.762dB)

(e) TV+new constraint (28.21dB)

Figure 2.4: CS recovered fishing boat image (30% measurements)

(a) Original

(b) 1-Norm (26.743dB)

(c) 1-Norm+new constraint (28.44dB)

(d) TV (24.914dB)

(e) TV+new constraint (26.816dB)

Figure 2.5: CS recovered Barbara image (40% measurements)

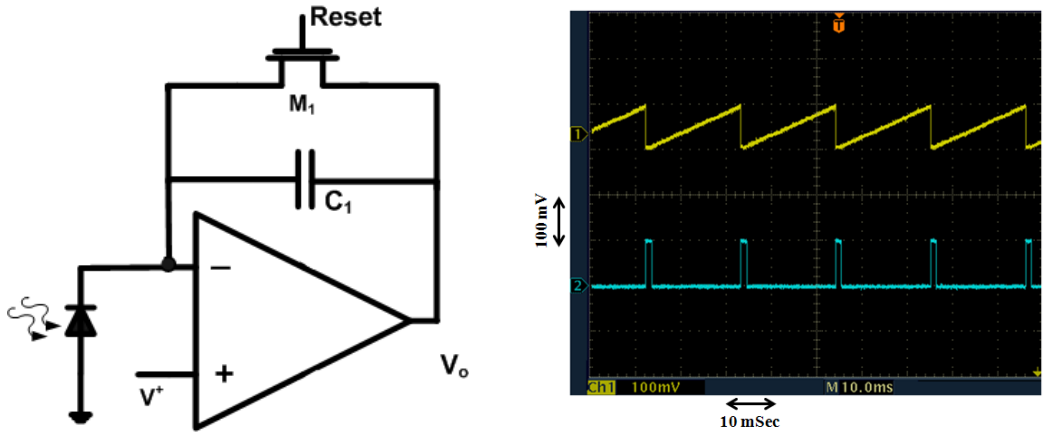
Chapter 3

Current-mode CS-CMOS imaging

3.1 Introduction

CMOS image sensors can be categorized into passive pixel sensors (PPS) and active pixel sensors (APS) [75]. In a PPS, the integrated charge is directly read out from the pixel and it is amplified outside the sensor array. In the case of an APS, in addition to the photo-sensitive components, active components are used to amplify the electrical signal related to the accumulated charge. Then, the amplified signal is read out from the pixel. In-pixel amplification improves the signal-to-noise ratio of the sensor. On the other hand, adding extra components to the pixel decreases its fill-factor.

Different APS structures such as three-transistor APS, APS with an integrator or APS with a comparator are discussed in [100, 101]. These structures differ in the way they amplify and convert charge to voltage and in the number of non-photosensitive components they use in the pixel. Compared to other two structures, the APS with an integrator is of highest linearity with respect to the incident light power. Figure 3.1 shows a schematic and the output waveform of the APS with an integrator.



(a) (b)

Figure 3.1: (a) Schematic of APS with integrator, (b) Measured waveform for the circuit. Channel 1 (upper trace) is the the measured output waveform and channel 2 (lower trace) is the reset signal applied to the pixel.

When the reset signal is high, the capacitor is discharged, and the output is equal to the DC voltage applied to the positive input of the operational amplifier. When the reset is low, i.e., during the integration time, the photocurrent is integrated in the feedback capacitor, C_1, while the DC voltage applied to the photodiode, V^+, is constant. Because of the constant voltage on photodiode sense node, the photocurrent is integrated in a constant capacitor, C_1, rather than the photodiode capacitor. This leads to the linear output. The output voltage is given by the following equation:

$$v_o(t) = V^+ + \frac{1}{C_1} I_D(t - t_0), \tag{3.1}$$

where t_0 is the beginning instant of integration interval and I_D is the photocurrent.

In this chapter the idea of APS with integrator is used to implement the CS measurement for an array of photodiodes, i.e., a sensor array.

3.2 Current-mode CS sensor

For hardware implementation of the CS method in electrical domain, binary random generator can be exploited to provide the random coefficients for different pixels of the array. When the measurement matrix is a random binary matrix, the measurement process is the addition of some randomly selected pixels for each measurement. Each measurement corresponds to one row of the measurement matrix, $\mathbf{\Phi}$. Therefore, the pixels corresponding to all 1's in each row of $\mathbf{\Phi}$ should be selected and the accumulated level of the light over all selected pixels should be considered as the CS measurement. Thus, the measurement process is performed in two steps: the selection of the pixels and then the extraction of the total light intensity.

3.2.1 Measurement technique

The linearity property of an APS with the integrator structure makes it suitable for designing a CS encoder. To implement the CS encoding by this structure, several photodiodes could be connected to the same integrator, as shown in Figure 3.2. Each photodiode, i.e., a pixel of the image, can contribute to the output voltage when its corresponding switch is on. Therefore for each measurement, switches corresponding to the 1's in one row of the measurement matrix, $\mathbf{\Phi}$, are on. The output for i^{th} measurement is given in (3.2).

$$v_{o_i}(t) = V^+ + \sum_{j|\phi_{ij}=1} \frac{1}{C_1} I_{D_j}(t - t_0). \tag{3.2}$$

To take advantage of the entire interval of integration, the output at the end of the integration time right before the next reset can be considered as the CS measurement value.

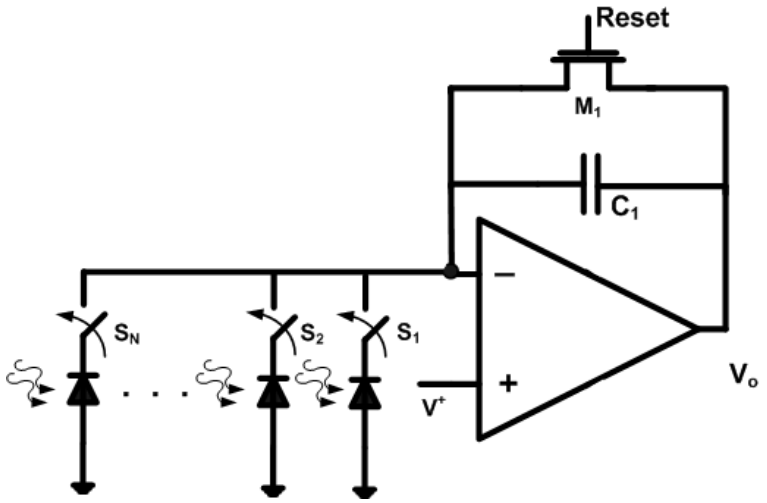


Figure 3.2: General schematic of the CS encoder.

Therefore, each integration time interval could be called as a measurement extraction interval. Also, during the integration time, the photocurrent of several photodiodes are added together rather than their corresponding output voltages. This is a current-mode implementation of CS for CMOS sensor arrays.

Figure 3.3 shows a schematic of a CS-CMOS imager for one pixel connected to the integrator. Other pixels can be connected to the negative input of the integrator which is shared among all pixels in the array. Each pixel consists of one photodiode and two transistors as current switches controlled by the signal V_s. Transistor M_1 provides the path for the photocurrent when the pixel is selected to contribute in the CS measurement. When V_s is high, M_1 (nMOS) is on, M_2 (pMOS) is off, and the photocurrent is integrated in C_2 during that time. When V_s is low, the incoming light to the pixel should have no effect on the measurement value. Therefore, transistor M_1 is off to prevent the photocurrent of the pixel from integrating in C_2. However, there might be leakage from the photocurrent even if the pixel is not selected for the measurement. Therefore, there should be a path to shunt the

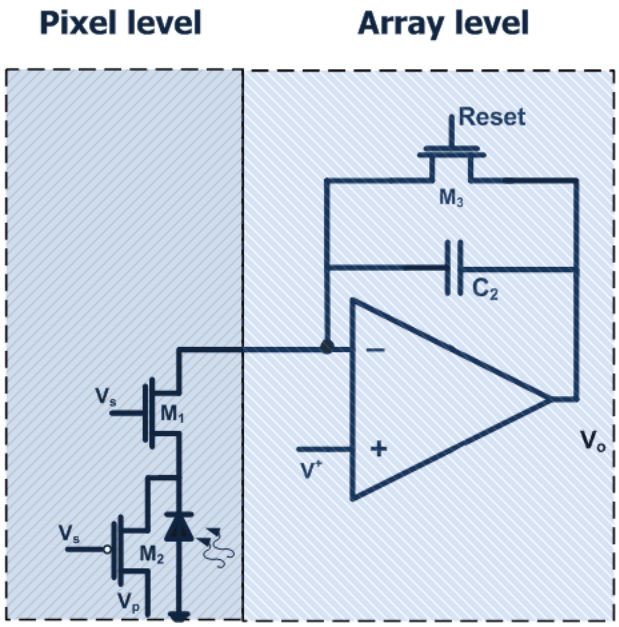


Figure 3.3: Schematic diagram of CS imager using APS with integrator structure.

photocurrent from leaking toward the integrator capacitor C_2. Transistor M_2 provides the shunting path for this unwanted photocurrent. The control voltage V_p, which is accessible from outside the chip, can set the V_{GS} of M_2 so that it shunts the unwanted photocurrent.

Figure 3.4 shows the simulation results for the CS measurement circuit in Figure 3.3 to evaluate the effect of the V_p in shunting the leakage current. Curve (1) shows the output when only one pixel is connected to the integrator, therefore the output corresponds to the integrated light in one pixel. Curve (4) shows the output for the situation when two pixel are connected to the integrator and both of them are selected for the CS measurement. Curve (2) and (3) show the situation when two pixels are connected to the integrator but only one of them is selected to contribute to the CS measurement. Since $V_p = 650mV$ for curve (2), then the leakage current from the non-selected pixel is shunted by M_2 in that pixel. The fact that curve (2) is almost the same as the curve (1) shows the effectiveness

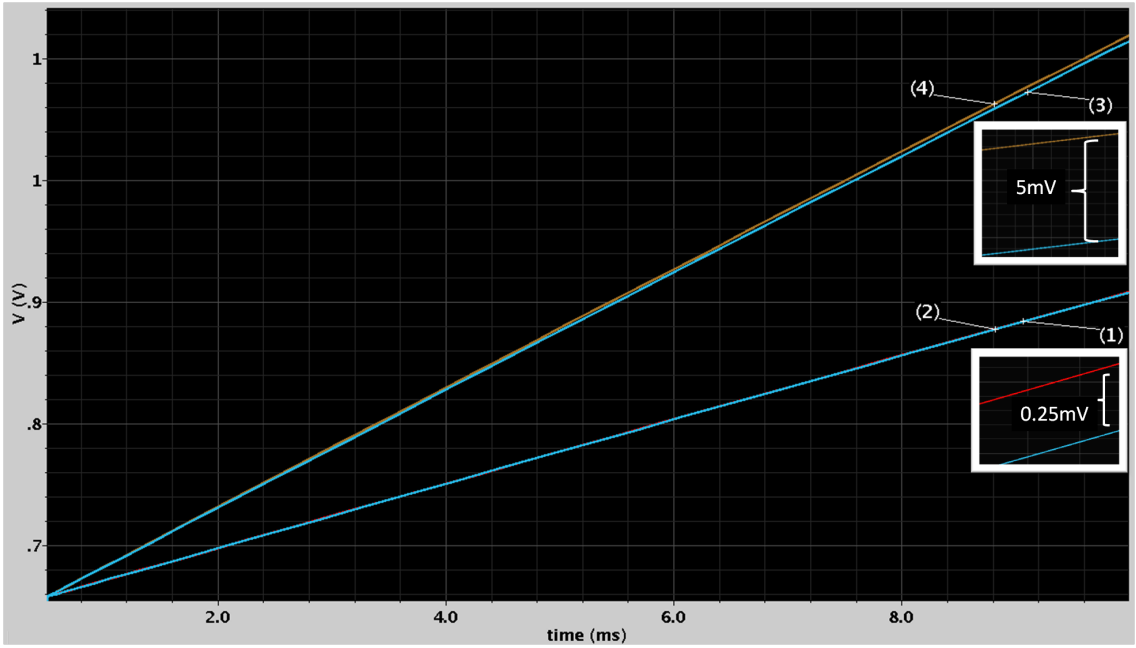


Figure 3.4: Simulations results for the output of integrator for $I_p = 10pA$ and $T_{int} = 10mSec$. (1) One pixel is connected to the integrator, (2) Two pixels are connected to integrator but one pixel is selected and $V_p = 650mV$, (3) Two pixels are connected to the integrator but one pixel is selected and $V_p = 0V$and (4) Two pixels are connected to integrator and both of them are selected.

of current leakage shunting process. For the curve (3), V_p is zero and there is no leakage shunting. The output is close to curve (4), and it means that without the leakage shunting a considerable amount of the photocurrent will leak toward the integrator and makes the selection process ineffective for the CS measurement.

3.2.2 Binary random generator

After connecting the photodiodes to the integrator, there should be some select signals to determine the state of the switch for each pixel and every measurement. In our design, linear feedback shift registers (LFSR) are used to provide pseudo-random, binary sequences for CS measurements. The LFSR is basically a shift register that advances the

signal through the subsequent flip-flops, bit by bit at each clock edge. The input bit of each shift register is a linear function of the outputs at the previous clock cycles. The linear function is implemented by using XOR gates and tapping the output signal at some of the flip-flops. By choosing an appropriate linear function, a proper pseudo-random pattern of 1's and 0's can be provided. For a LFSR with length n, the randomness is guaranteed if all $2^n - 1$ possible sequences, i.e., all sequences except an all-zero sequence, are sequentially produced once in $2^n - 1$ subsequent clock periods. In this case, the LFSR is called maximum-cycle LFSR. For each n, there are some specific positions for the taps which lead to the maximum-cycle property. We designed one of the maximum-cycle realizations of 16-bit LFSR.

The output of the LFSR at each clock edge represents one row in the measurement matrix $\mathbf{\Phi}$. Also, each register corresponds to a column in the measurement matrix and should be connected to one pixel to control its switch at different clock edges for different measurements. Therefore, to implement the CS encoder for an image with n pixels, a LFSR with n registers is required and the output of each register should be connected to one pixel as its *select* signal.

The LFSR could be designed outside the imager. But to feed the random sequence to the imager, a huge number of pins or some extra internal circuits are required. Therefore, it is more practical to design the LFSR inside the imager. Figure 3.5 shows the schematic of the 16-bit LFSR with 6 feedback taps. The positions of the taps, i.e., XOR gates, lead to the production of a maximum-cycle LFSR with up to $2^{16} - 1$ different bit sequences, thus assuring the pseudo-random property of the LFSR output.

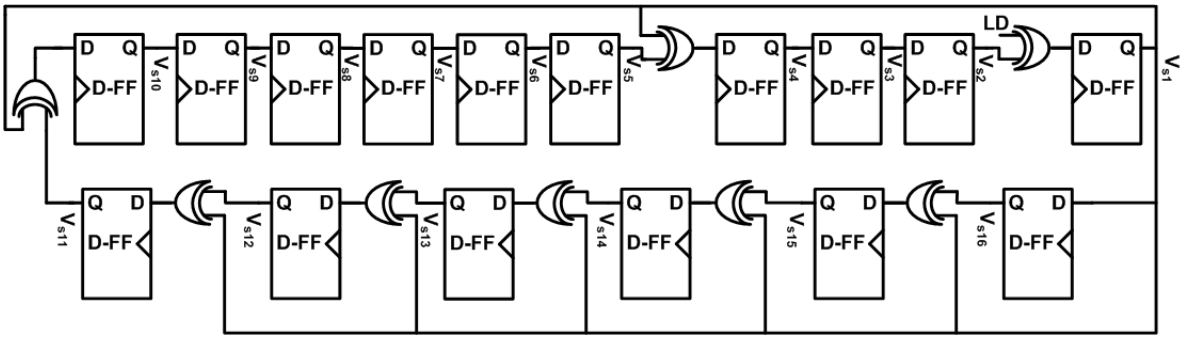


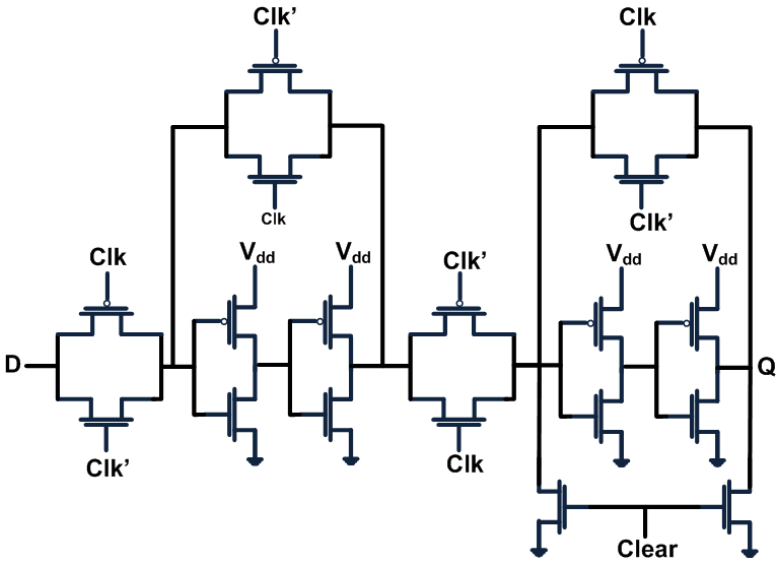
Figure 3.5: Schematic of the 16-bit LFSR.

Figure 3.6 shows the schematic of the internal circuits of (a) D flip-flop and (b) XOR gate. The dynamic structure of the flip-flop allows the possibility of using a high frequency clock. Also, the input and output latches guarantee the functionality of the LFSR in the case of delayed clock signals for different flip-flops.

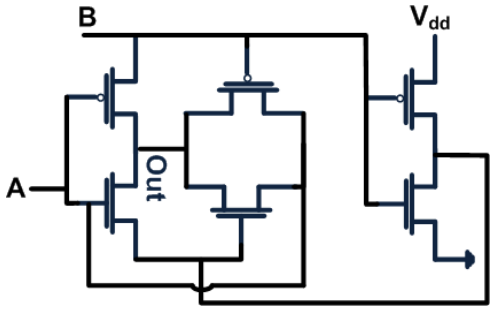
Figure 3.7 shows simulation results for the control signals and the output of the LFSR for three flip-flops, i.e., three random bits. Using the *LFSR clear* signal, the outputs of the all flip-flops in LFSR become zero at the first rising edge of the *LFSR clock*. At the second clock rising edge, *LFSR load* signal is high and the first register, Q_1 is loaded by one while the other registers are zero, i.e., the initial seed of the LFSR. After the second edge, the LFSR produces one new random sequence per each clock period.

3.2.3 Block coding

In our implementation of a CS-CMOS image sensor, the CS measurement process is performed for separate blocks instead of the whole sensor array at the same time. The following practical issues lead to the use of block by block CS encoding instead of the implementation for the entire array at the same time.



(a)



(b)

Figure 3.6: Schematic of the integrated (a) D flip-flop and (b) XOR gate.

1) LFSR size. Considering an imager with large array size and if the CS encoding is implemented for the entire imager at the same time, there should be a large LFSR on chip to provide the *select* signals for all the pixels at the same time. In this case, the size of the LFSR is proportional to the size of the array. Therefore, for imagers with large array sizes, a considerable amount of the imager area is used for the LFSR. On the other hand, this area could be used to increase the number of pixels, thus improving the spatial resolution of the imager.

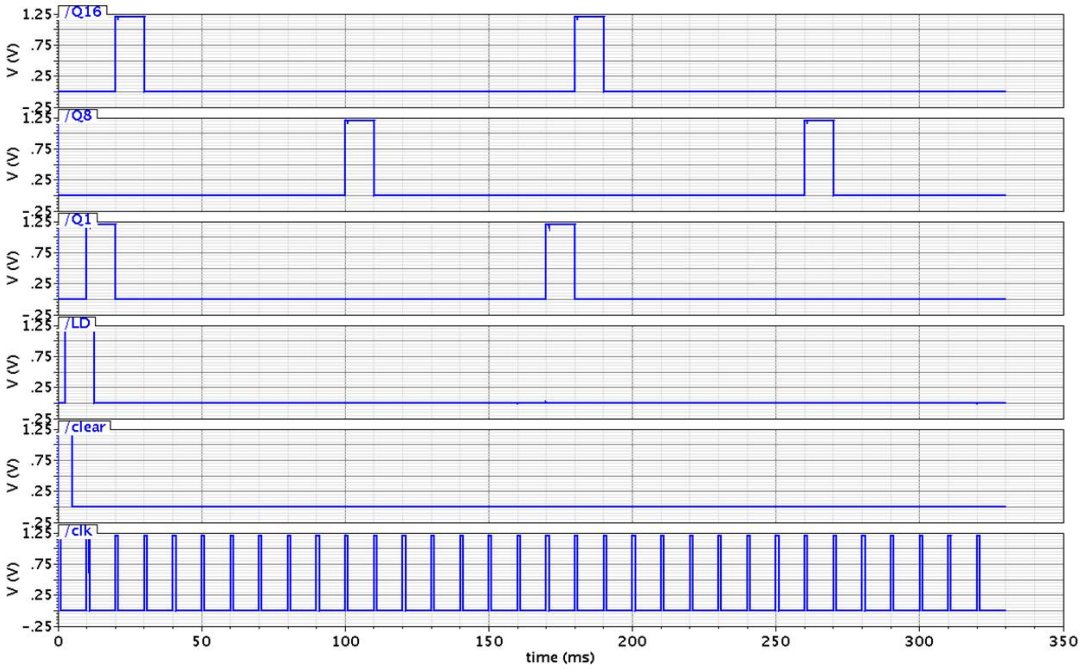


Figure 3.7: Simulation results for the designed LFSR. $/Q_{16}$, $/Q_8$, and $/Q_1$ are 3 bits of the LFSR output. /LD, /clear, and /clk are the *LFSR load*, *LFSR clear* and *LFSR clock*, respectively.

2) Layout restrictions. The LFSR circuit should be placed outside the sensor array to keep the fill-factor of the imager as high as possible. Therefore, there should be a separate metal path from the array toward the LFSR for each pixel. Also, these metal paths cannot pass over the photo-sensitive area of pixels. Thus, to connect the sensor array to the LFSR and to provide the individual access to each pixel, several parallel, metal paths are required to be between the different columns of the sensor array. These parallel paths degrade the spatial resolution of the imager. By increasing the size of the imager, the number of required parallel paths increases, and it leads to more degradation in the spatial resolution. Also, there is a limited number of metal layers in each technology. Therefore, it is impractical to make the connection between pixels and LFSR for an imager of large array size.

3) Scalability. To extend the imager to any desired size, the design of the imager should be scalable for any size of the imager. In the case of implementing the CS encoding for the entire array at the same time, the entire layout should be remade for any new size, which means that the design is not scalable. However, by using the block coding scheme, any array with desired size can be made by stitching different blocks together.

4) Reconstruction time. The CS decoding algorithms are usually costly in terms of computational complexity. To make the decoding process faster, the image is divided into different blocks and the CS compression is performed for separate blocks instead of the entire image at the same time. By using this block strategy, the total time spent for the decoding of all blocks is less than the time required to decode the entire image at the same time.

We considered the drawbacks of using a very large LFSR inside the chip, layout restrictions for connecting the array to the LFSR, the scalability of the imager and the fact that decoding would be performed for separate blocks. Therefore, in our implementation, the CS encoding is performed for separate 4×4 blocks of the image.

Figure 3.8 shows a 4×4 block of pixels and the interconnections. To make the CS measurements for each block, 16 *select* signals should be connected to the LFSR outputs, V_{s_1} to $V_{s_{16}}$ in Figure 3.5. Also, the CS encoding is performed for different blocks of the image one by one. The integrator and capacitor C_2 are shared among different blocks. Therefore, among all of the blocks which are connected to the same integrator, one block can be measured at the time. Thus, there is a *block switch* for each block to control the connection of the block with the integrator. However, there might be a leakage current from the non-selected blocks towards the integrator. The structure of the *block switch* allows for minimizing the effect of the leakage current by calibrating the V_p voltage. Transistors M_4 and M_5 are working as the current switch for each block and their functionality are the same as transistors M_1 and

M_2 inside the pixels.

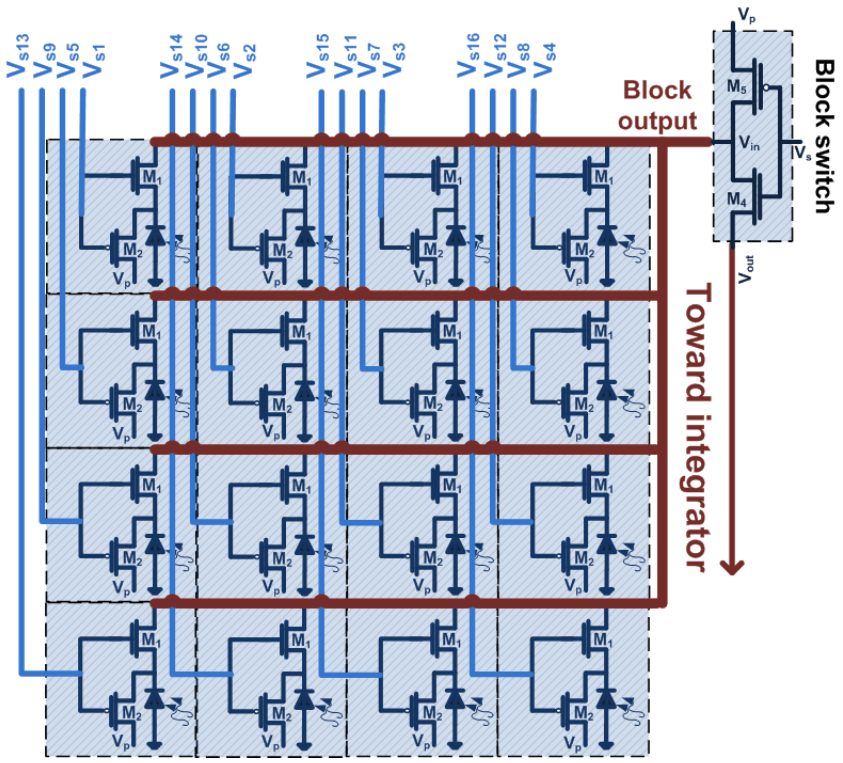


Figure 3.8: A 4×4 block of pixels with their interconnections.

3.2.4 Sensor architecture

Conventional image sensors use row-select transistors and column read-out circuits to read the value of all pixels so that the value of each pixel is individually accessible. The output of the pixels in the same column are connected together and there is one signal path per column from the pixels' outputs to the read-out circuits outside the array. Therefore, sharing the data path among all pixels in one column instead of one data path for each pixel, guarantees the feasibility and scalability of the imager design for any desired array size.

The read-out of the array could be performed using a pixel-by-pixel (PBP) or an array-level (AL) read-out strategy [102]. For PBP read-out, only one analog-to-digital converter (ADC) for the entire array is needed. Therefore, the total read-out time for each frame of an $n_1 \times n_2$ array using PBP read-out, T_{PBP}, is

$$T_{PBP} = (n_1 \times n_2)(T_{A/D} + T_O). \tag{3.3}$$

where $T_{A/D}$ is the conversion time of ADC for each pixel value and T_O is the time taken to send the converted digital result out of the imager.

In the AL read-out scheme, there is one ADC for each column of the array. The reset signals are activated one by one for all rows, and the pixel values are read out one row at a time [75]. This is the same technique that is used in the electronic rolling shutter. This parallel strategy for the read-out improves the read-out time for each frame, and using AL read-out, T_{AL} is

$$T_{AL} = n_1 T_{A/D} + (n_1 \times n_2) T_O. \tag{3.4}$$

For both PBP and AL schemes, the imager is spatially scalable for any desired array size with no extra complication in hardware usage or layout design inside the array. However, the size of the read-out circuits changes for different array sizes in the AL structure. The frame rate decreases in proportion to the number of pixels in the array for PBP structure and number of rows for AL structure. Therefore, AL structure is preferred to PBP in terms of frame rate for high resolution sensor arrays because of its superior read-out speed.

In the CS image sensor, since the value of each pixel is not separately acquired, the read-out circuits should be designed to properly read the values of the CS measurements for different blocks. All blocks in the array share a same LFSR to take the required random

select signals. Therefore, they share the same paths to connect their *select* signals to the LFSR. Figure 3.9 shows the connection of 16 different 4×4 blocks and their metal paths towards the LFSR. Each path shown in the figure contains a pack of 4 different metal layers for all of the pixels in one column. As there are 16 pixels in each block, 4 parallel metal packs pass through each block. Also, the metal packs for different blocks are connected together and provide the connection of the array to the LFSR. Therefore, all of the blocks are always connected to the LFSR, but only the blocks with activated *block switch* shown in Figure 3.8 are involved in measurement extraction.

For the read-out of the array and connections of pixels to the integrator, the PBP or AL schemes can be used. However, in the case of a CS-CMOS imager, there are blocks of pixels instead of individual pixels. Also, the measurement values should be read out instead of the pixel values. By using one integrator for the whole array, one block could be read out at a time. Figure 3.10 shows the block-by-block (BB) read out scheme for the CS imager. The read-out process is performed for the blocks one by one. This is the same as for conventional imagers mentioned before, but instead of reading the level of the light pixel by pixel, the aggregate levels of the light, i.e., measurement values, are read out block by block, i.e., block-wise electronic rolling shutter. To select a block for the measurement read-out, row addressing circuit activate the *block switch* of all blocks in one row, but only one of them is selected to be connected to the integrator using the column addressing circuits.

In the case of block-by-block (BB) read-out, the read-out time for each frame for an $n_1 \times n_2$ block and M measurements per block is proportional to the number of pixels (see

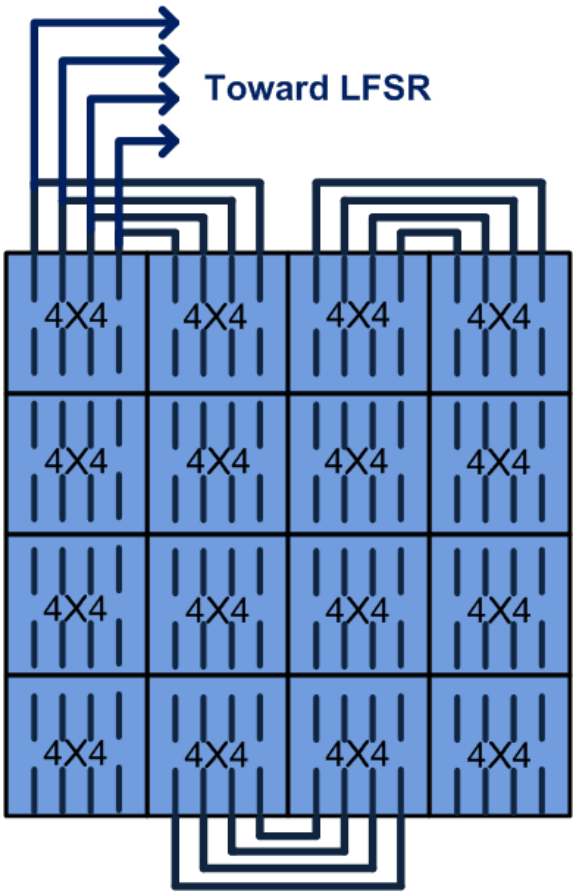


Figure 3.9: Connection of 16 blocks to the same LFSR.

(3.5) below for the BB read-out time, T_{BB}).

$$T_{BB} = M\frac{(n_1 \times n_2)}{16}(T_{A/D} + T_O). \tag{3.5}$$

To improve the read-out time for large arrays, columns of blocks could be considered for parallel read-out instead of columns of pixels in AL read-out. All blocks in each row can be read out at the same time by using one integrator per column of blocks. Figure 3.11 shows the AL read-out structure implemented for the CS-CMOS imager, i.e., column-of-block

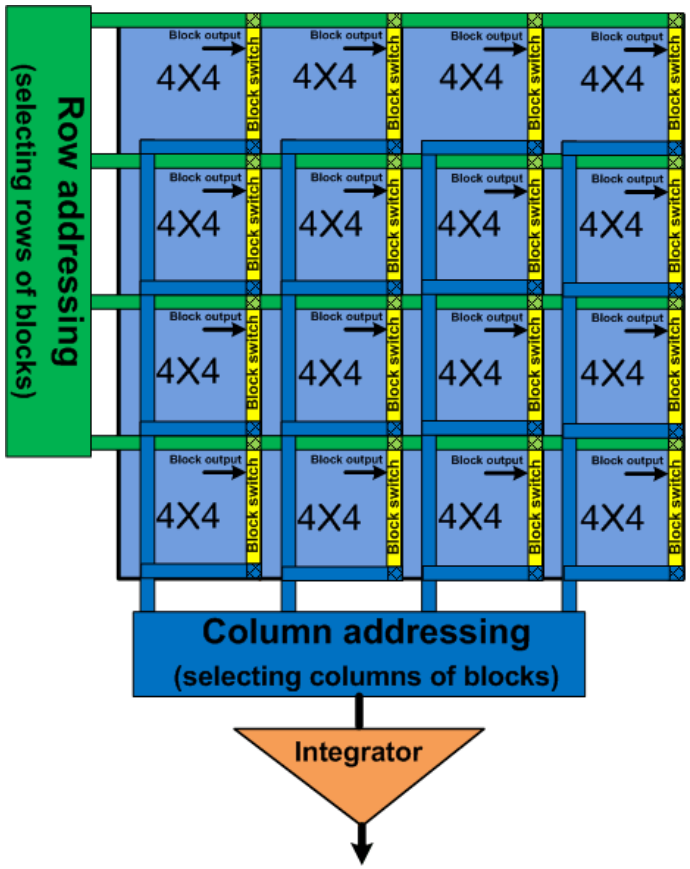


Figure 3.10: 16 blocks with interconnections for block-by-block (BB) read-out.

(CB) structure. By using 4 integrators, the CS measurement process is performed for all blocks in each row at the same time, while the row addressing circuits activates the block switches for the blocks in that row. Therefore, according to (3.6) below which gives the CB read-out time, T_{CB}, the read-out time for each pixel is proportional to the number of rows instead of the number of pixels. This makes the CB structure more scalable in time domain than the BB scheme.

$$T_{CB} = M\frac{n_1}{4}T_{A/D} + M\frac{(n_1 \times n_2)}{16}T_O. \tag{3.6}$$

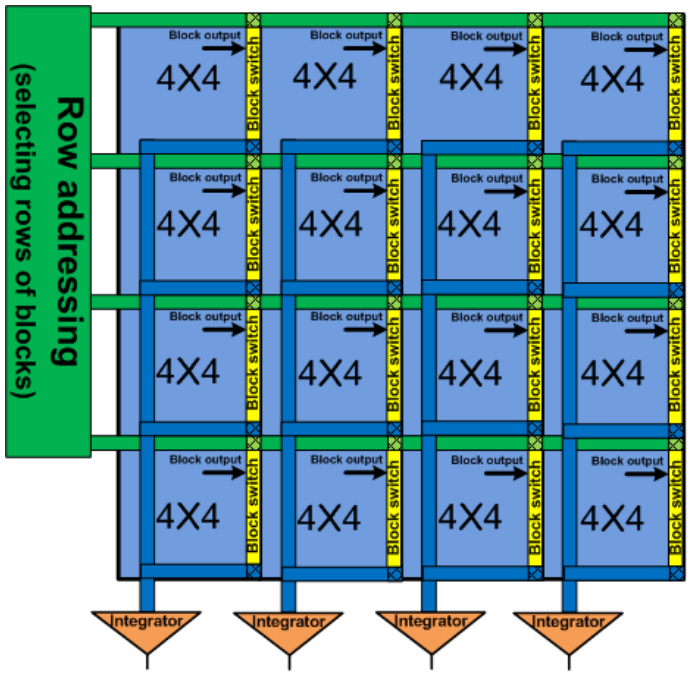


Figure 3.11: 16 blocks with interconnections for column-of-block (CB) read-out.

3.3 Experimental results and discussions

The current-mode structure was designed and fabricated in a commercial 130nm CMOS technology for testing the functionality of the CS measurement process. The chip was fabricated and packaged through CMC microsystems. The design was tested for low-level light illumination. Therefore, a large photodiode was used to achieve high sensitivity for low-level light applications. The chip was mounted on a PCB board which is designed using Cadence Allegro. The PCB board includes the chip package, a discrete A/D converter and the peripheral circuits to make the connection to the FPGA board. An Altera NIOS II development kit was used to provide the required clocks for the chip and also to read the digital output of the A/D converter using a serial interface. Also, different BNC connectors

were used to provide the access to the analog output of the chip. Figure 3.12 shows the general structure of the measurement set-up. All measurements were performed in an optically dark room to minimize the background illumination. A high quality Fiber-lite illuminator was used to provide different levels of the light for the measurements. Also, the power of the light was measured with a Newport calibrated power meter with its optical sensor is positioned on the optical table at the same place as the chip.

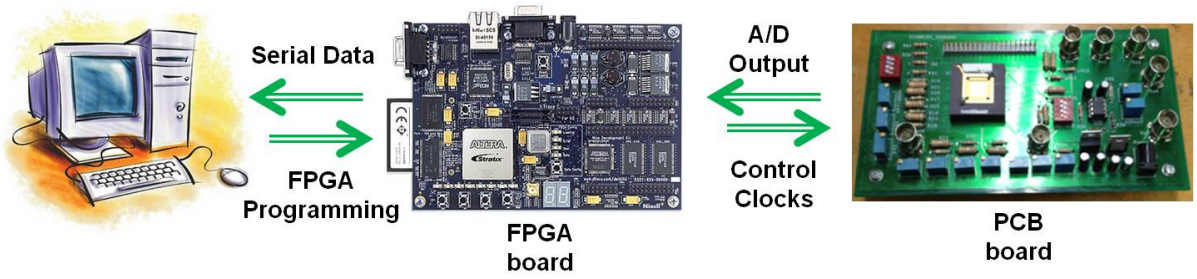


Figure 3.12: General structure of the measurement set-up.

3.3.1 16×16 array

A 16×16 array has been implemented to evaluate the feasibility of the block coding scheme. Figure 3.13 shows the layout for a 16×16 array. Here, 16 blocks are connected together and they share one integrator and a 16-stage LFSR circuit. As shown in the figure, all blocks are connected together, similar to the blocks in Figure 3.9. This method of connection can be used to build imagers with any desired size using 4×4 blocks. Since the measurement process was verified for any 4×4 block, and the block-by-block read-out works for all 16 blocks, then the feasibility of the design for larger array sizes is confirmed. Note that in our test structure the block-by-block read-out with one integrator is considered

which is simply scalable to column-of-blocks read-out by adding more integrators.

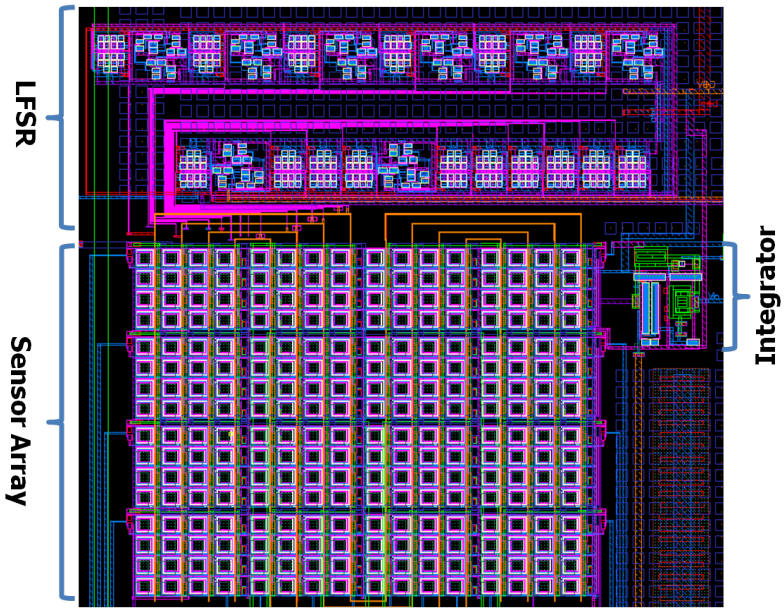


Figure 3.13: Layout for a 16×16 imager

3.3.2 2×2 block

To validate the functionality of the proposed current-mode, CS-CMOS measurement circuit, a 2×2 block was designed and fabricated in a commercial 130nm CMOS technology. For the 2×2 block, 4 pixels are connected to the integrator. Therefore, a *select* sequence with 4 bits is used to select some of the pixels to integrate their photocurrent in C_2. Different sequences lead to extraction of different CS measurement values for the array. Figure 3.14 shows the oscilloscope's screen shot for the reset and output signals for four different *select* sequences and the same level of light. As shown in the figure, the output waveform is rising with a higher slope by adding more pixels together, i.e., more 1's in the *select* sequence. By sampling the output right before the next reset signal, a correct measurement

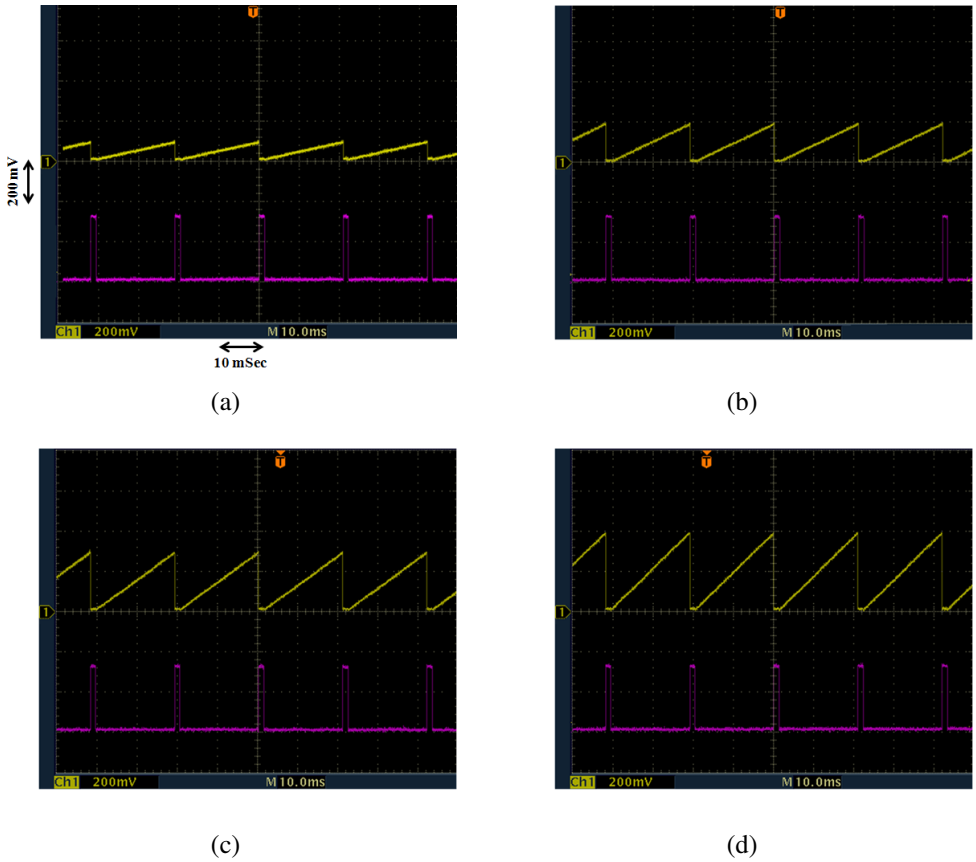


Figure 3.14: Output (upper traces) for a 2×2 block and reset signal (lower traces). (a) One pixel selected, (b) Two pixels selected, (c) Three pixels selected, and (d) Four pixels selected.

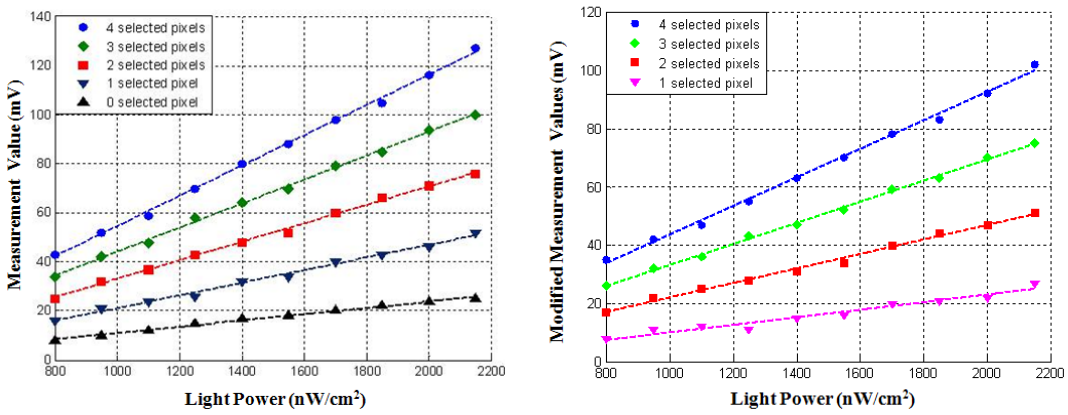
value which is proportional to the level of the light and the number of effective pixels is obtained. Note that the difference between the sampled value and V^+ is considered as the measurement value.

To investigate the linearity and monotonicity of the measurement method, Figure 3.15(a) shows the trend of measurement values with respect to the levels of the light for different *select* sequences. For each curve, the measurement value increases almost linearly with the level of the light. Also, for each level of light, there is a monotonic relation between the

measurement values and number of selected pixels. In the case of 0 selected pixel, theoretically, the level of the light should not affect the measurement values. However, because of current leakage from the pixels, even when their switches are off, there is a slight increase in measurement values by raising the level of the light. Note that transistor M_2 suppresses the leakage current, but it cannot be completely removed. Therefore, there is an error, Δm, in the measurement values captured for different number of selected pixels. The values for the case of 0 selected pixel show the amount of the measurement error. Therefore, the curve for 0 selected pixel is the variation of Δm with the level of light. This error can be subtracted from the captured measurement values to provide more accurate values. Figure 3.15(b) shows the trend of the measurement values for different number of selected pixel after modifying the effect of Δm. The slopes of the curves in Figure 3.15(a) are 0.0617, 0.0489, 0.0376 and 0.0257 for 4,3,2 and 1 selected pixels, respectively. Because of the measurement error, the slopes are not properly proportional to the number of selected pixels. However, the slopes of the curves in Figure 3.15(b) are 0.0489, 0.0362, 0.0248 and 0.0128 for 4,3,2 and 1 selected pixels, respectively. It can be seen that the slopes are more proportional after the modification of measurement error.

3.3.3 4×4 block

In our design, the imager is built of several 4×4 blocks. Therefore, a 4×4 block has been designed and fabricated to verify the validity of the measurement values for each block. The *select* signals of the block are connected to the outputs of a 16-bit LFSR fabricated on-chip. Figure 3.16 shows the control signals for the LFSR and the reset signal. The first two clock cycles are for the clearing the flip-flops and loading the initial seed for the LFSR.



(a) (b)

Figure 3.15: Measurement values for different levels of light, 2×2 block and different number of selected pixels. (a) Measured values, and (b) Modified values after subtracting the measurement error Δm.

M measurement values are made for the first block of the image after the M following integration periods. As shown in Figure 3.16, for each set of M following reset periods, M CS measurements are produced for a block or column of blocks of the image. The LFSR makes up to $2^{16} - 1$ different sequences, thus leading to the different measurement vectors for different blocks. Note that the initial seed for the LFSR is "1000000000000000". To have a different initial seeds more than two clock cycles is required to get the desired seed before the CS measurement begins.

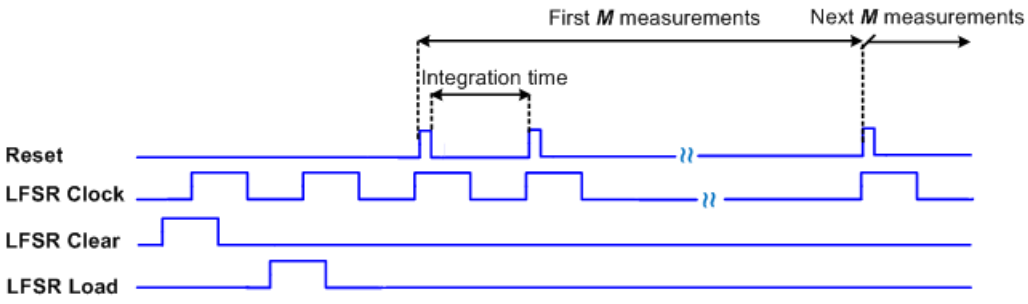
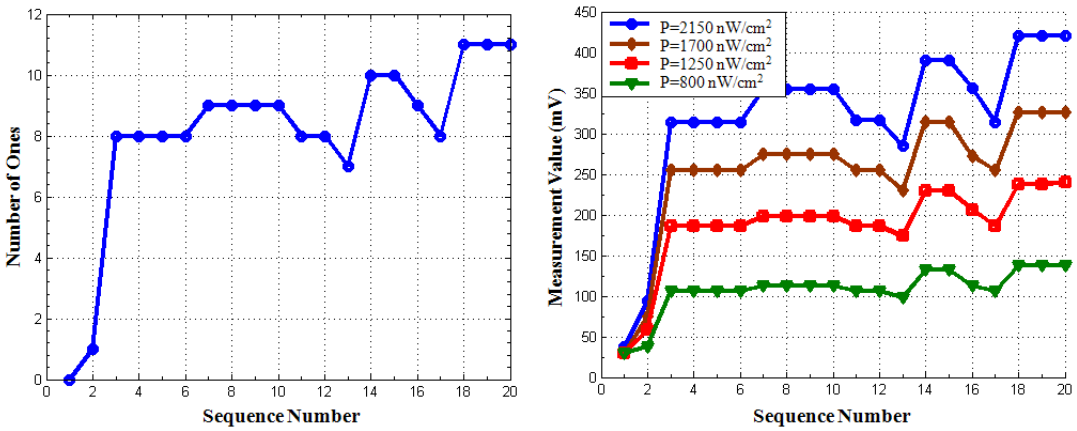


Figure 3.16: Timing diagram for the reset and LFSR control signals.

Figure 3.17(a) shows the number of ones in the first 20 sequences of LFSR, e.g., the third sequence has 8 one bits out of its 16 bits. Since the number of ones reflects the number of pixels which contribute to the measurement process, then for uniform illumination, the measurement values should have the same trend as the number of ones in the random sequences. Figure 3.17(b) shows the measurement values with respect to the number of ones in their corresponding sequences for four different levels of light. As can be seen in the graph, the non-monotonic trend for each curve is similar to Figure 3.17(a), although the measurement values are proportional to the level of the light.



(a) (b)

Figure 3.17: (a) Number of ones in the first 20 sequences of LFSR output. (b) Measurement values correspond to first 20 sequences for different levels of light.

Figure 3.18 shows the osilloscope's screen shot for the output of a 4×4 block, reset signal, LFSR load signal and LFSR clear signal. As shown in the figure, the LFSR clear signal is high for two periods of integration time. During this time, none of the pixels are selected for the measurement, although there is a small rise in the output because of the leakage current from non-selected pixels. The amount of this rise can be measured and used to

compensate the effect of the leakage current for other measurement values. There is a one in the *select* sequence for the next integration period when the load signal in high and one pixel is selected. Other integration periods are for the *select* sequences with different number of ones based on Figure 3.17(a). It can be seen that the amount of the rise in each integration period is proportional to the number of selected pixels.

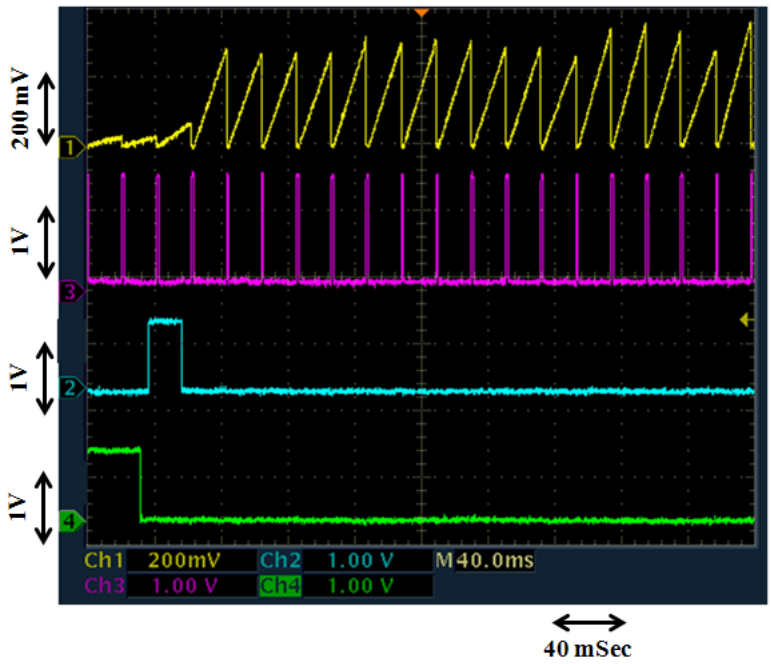


Figure 3.18: Output for a 4×4 block (channel 1), reset signal (channel 3), LFSR load signal (channel 2) and LFSR clear signal (channel 4).

Figure 3.19 shows the measurements values versus the level of the light for four different sequences. The measurements values linearly change with the level of the light with slopes of 0.223, 0.196, 0.175 and 0.154 for 11, 10, 9, and 8 selected pixels, respectively. However, as can be seen in figure 3.18, when none of the pixel are selected for the measurement there is still a rise in integrator output. This rise is the measurement error, Δm, which should be

measured and compensated to modify the measurement values. Figure 3.20(a) shows the measurement values with the number of selected pixels for 5 different levels of light. The measurement value when no pixel is selected shows the amount of the error for each level of light. Also, Figure 3.20(b) shows the amount of the error with the level of the light. The linear increase in the error with the level of the light, shows the dependency of the error with photocurrent which is because of the leakage current form non-selected pixels.

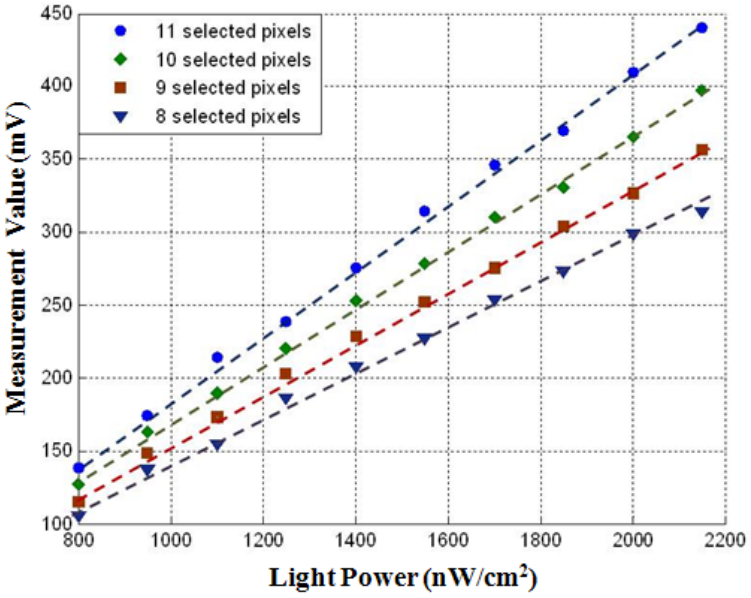
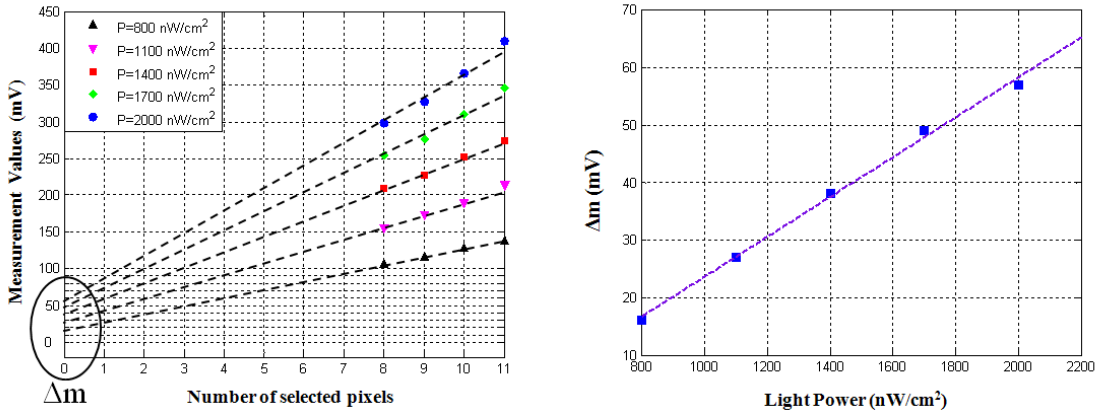


Figure 3.19: Measurement values for different levels of light, 4×4 block and different number of selected pixels.

Figure 3.21 shows the measured signal-to-noise ratio (SNR) of the measurement values for different *select* sequences and different levels of the light. The SNR values was extracted by multiple measurements of the output signal for each sequence and each level of the light. About 60 to 120 voltage samples are collected at each illumination level. The mean was



(a) (b)

Figure 3.20: (a) Measurement values for different number of selected pixels, 4×4 block and different levels of light. The values for zero selected pixel show the amount of the measurement error, Δm, for each level of light. (b) Δm for different levels of light.

considered as the signal value and the standard deviation as the noise. Note that the SNR was evaluated for the swing signal which is the difference between the voltage values at the end and the beginning of the integration time, i.e., measurement values.

The measurement results in Figure 3.21 show that the SNR increases with the level of the light. Also, by increasing the number of the selected pixels, the SNR increases for the same level of the light. It is because the power of the signal, i.e., measurement value, depends on the level of the light and number of the selected pixels. Therefore, for same level of the light, the SNR value is higher when more pixels are added together. However, the difference between the curves for different number of selected pixels is higher for lower level of light and decreases for higher light level. It is because of the the dominance of different sources of noise for different light levels.

The main sources of the noise can be categorized to read-out noise and shot noise. The average noise power of the read-out noise is independent of the light level while it increases

with photocurrent for shot noise. Therefore, for higher level of the light, i.e., higher photocurrent value, the dominant source of noise is the shot noise. Also, by adding more pixels together, more sources of shot noise affect the value of the CS measurement. Therefore, since the shot noise is more effective in the case of higher level of light and higher number of selected pixels, the differences between different curves in Figure 3.21 decrease by increasing the level of light.

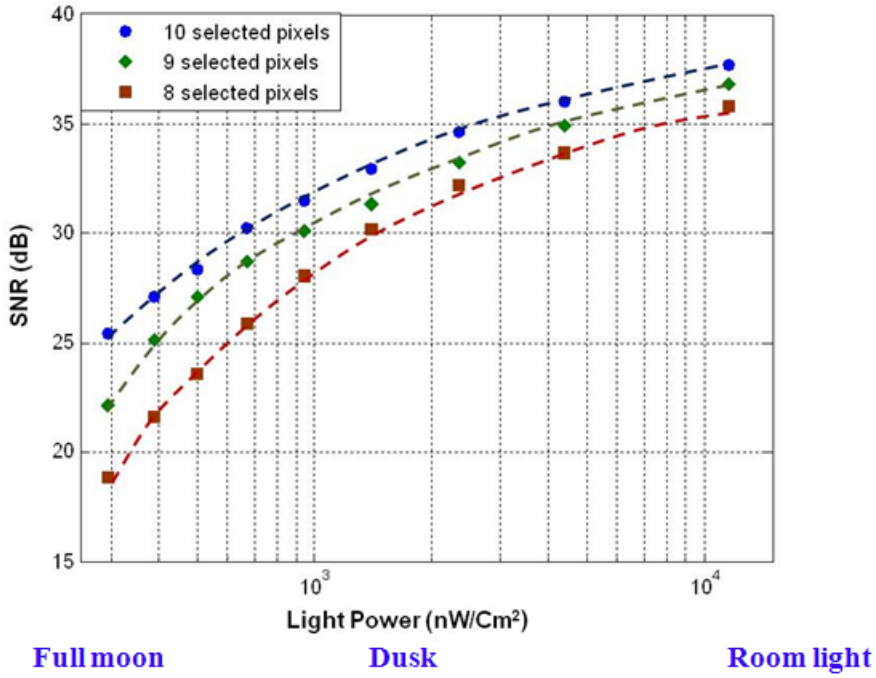


Figure 3.21: Signal-to-noise ratio (SNR) for different levels of the light and different number of selected pixels.

3.3.4 Image Reconstruction

It is not very useful to take images of natural scenes by using a small 16×16 array due to the low spatial resolution. However, small image patches could be used to evaluate the reconstruction performance of the imager. Figure 3.22 shows the reconstructed images for two small image patches of letters 'X' and 'D' for several compression ratios, i.e., $\frac{M}{16}$. The size of each image patch is 32×32. Each patch has been divided to four 16×16 sub-patches. The CS measurements have been extracted from the chip for each sub-patch separately. The extracted measurement values have been used to perform the reconstruction for each sub-patch and the reconstructed sub-patches have been stitched together to make the final reconstructed patch for each image. As shown in Figure 3.22, the quality of the reconstructed image is reduced when the number of measurements decreases. Therefore, there is a trade-off between the quality of each frame of the image and the amount of data which is sent off the chip for that frame.

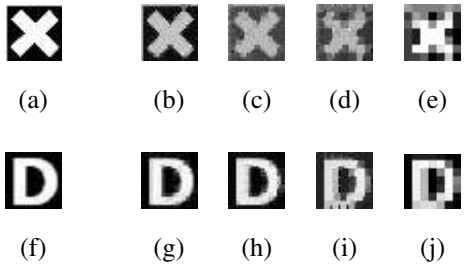


Figure 3.22: Image of two samples and their reconstructed versions for different number of measurements, (a) and (f) Original images; (b) and (g) M=12; (c) and (h) M=8; (d) and (i) M=4; (e) and (j) M=1.

To investigate the effect of the noisy measurements in the reconstruction quality, Figure 3.23 shows the simulation results for the Cameraman image and different numbers of noisy and non-noisy measurements. The amount of the additive noise for each measurement was

extracted from the SNR graph. $PSNR$ values as the mean square of the reconstruction error, are used to numerically evaluate the quality of the image reconstruction. Numerical and visual evaluation of the image quality with respect to the number of measurements are shown in Figure 3.23. It can be seen that noisy measurements decrease the quality of the reconstruction, although this degradation is less noticeable for the lower number of measurements. It is because for the higher level of the compression, the reconstruction error is more dominant compared to the level of the noise in measurement values.

(a)

(b) PSNR=29.05 dB

(c) PSNR=25.03 dB

(d) PSNR=23 dB

(e) PSNR=22.12 dB

(f) PSNR=20.10 dB

(g) PSNR=19.8 dB

Figure 3.23: Cameraman image and its reconstructed versions for different number of measurements with and without noise, (a) Original image, (b) M=12, (c) M=12 (Noisy measurements), (d) M=8, (e) M=8 (Noisy measurements), (f) M=4, and (g) M=4 (Noisy measurements).

Chapter 4

Voltage-mode CS-CMOS imaging

4.1 Introduction

In the previous chapter, the APS with integrator structure was used to implement a current-mode CS encoding structure. However, to increase the fill-factor of the imager, the integrator part was moved outside the array. Therefore, as the amplification is performed outside the pixel, the noise performance and the sensitivity of the system is inferior to conventional imagers with one integrator in each pixel.

In this chapter, a voltage-mode CS measurement structure is presented. This structure uses a 3-transistor APS and switched capacitor (SC) circuits as the analog adder. The internal structure of the APS pixels is similar to the pixel structure in conventional imagers. Therefore, the charge amplification is performed inside the pixel to keep the noise performance and sensitivity of the pixel. The output voltage is used by the analog adder to make the CS measurement values during the image acquisition. Since the measurement value is ready at the end of the integration time, then there is no need to spend extra time on top of the integration time for the addition, so the process is real-time. The switched capacitor circuits are off the sensor array and can be shared between multiple blocks. Also, by exploiting a

block read-out scheme, the design is scalable and could be extended for imagers of desired array size while the fill-factor and sensitivity of the imager remain unchanged.

4.2 APS with analog adder

4.2.1 3-transistor APS structure

The conventional APS structure with three transistors (3T-APS) is the simplest and most commonly-used APS structure. It also has the highest fill-factor among other APS structures. Figure 4.1 shows the schematic for this APS circuit. This structure is very suitable for low noise and high resolution applications [100, 101].

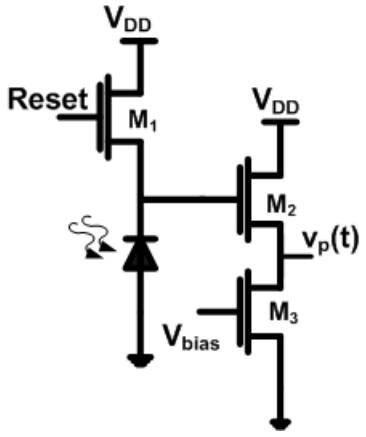


Figure 4.1: 3-transistor APS

In a 3T-APS, at the beginning, the reset signal is high and M_1 turns on, i.e., the reset period, and the photodiode's capacitive junction is being charged. When the reset signal is off, i.e., during the integration time, the junction is being discharged and the output voltage drops in proportion to the level of the incoming light on the photodiode. The value of the

output voltage at each instance and before saturation, shows the amount of light which is integrated in the pixel up to that instant.

To take advantage of the whole integration period, the value of the output at the end of the integration time and before the next reset should be measured to appropriately show the level of the light on the corresponding pixel. The lower the voltage at this instance, the higher is the incoming light intensity.

The output voltage of each pixel is given by:

$$v_p(t) = v_{o_{RS}} - \frac{i_{PH}}{C_{PH}} t, \tag{4.1}$$

where, $v_{o_{RS}}$ is the output voltage at the end of the reset period, i_{PH} is the photocurrent and C_{PH} is the junction capacitance of the photodiode. Note that the effect of dark current is neglected. Figure 4.2 shows the output voltage of the designed APS for the integration time of 320μsec. This APS was designed and fabricated in a $130nm$ CMOS technology.

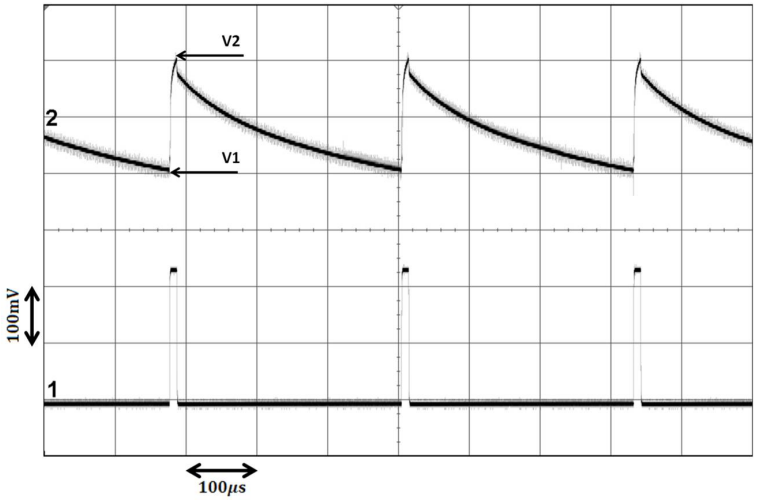


Figure 4.2: Reset signal on channel 1 and APS output for integration time of 320μsec on channel 2 ($V_2 - V_1 = 200mV$). The y-axis represent channels 1 and 2 voltages and the x-axis is time.

4.2.2 Analog adder

When the measurement matrix is a random binary matrix, the process of CS measurement becomes the addition of the outputs of some randomly selected pixels. The result of this addition can be used as the compressed image in the image decoding, which will be performed outside the imager.

The pixel values can be added together outside the sensor array. However, this requires all pixels to be completely read out and fed to the analog-to-digital convertor and subsequently the adder circuit. In this case, the compression process improves the throughput of the camera because for each frame, M measurement values instead of the N pixel values are sent off the chip and $M < N$. However, the throughput of the array remains the same as the imager without CS since N pixels should be read out and fed to the adder circuit. In other words, the read-out of the array takes the same time as the read-out time of a regular imager with no compression. Thus, although we have bandwidth saving because of the higher camera throughput, the compression results in no improvement in the array read-out time and frame rate. Also, reading all of the pixels one by one and using digital adders requires the use of off-array or even possibly in-pixel memories to store the values. Therefore, this read-out process requires more hardware, thus reducing the fill-factor of the imager.

To mitigate the problem of non-optimal array read-out and to make the measurement process less costly in terms of time, fill-factor and hardware usage, we propose the addition to be performed in analog domain. Instead of reading and saving the pixel values and subsequently adding them to form the CS measurements, our scheme calculates the CS measurements directly during the integration time. Therefore, our implementation only needs the read-out time for M measurements for each frame, while in the case of using the a digital adder outside the array, three time periods are required:

(1) reading and analog-to-digital conversion of N pixel values,

(2) adding the values together, and

(3) reading M measurements out of the chip.

For analog addition, we used switched capacitor (SC) circuits to add the pixel outputs together and to find the aggregate light level as the CS measurement. Figure 4.3, shows the conventional schematic view of a simple switched capacitor circuit. ϕ_1 and ϕ_2 are two non-overlapping clocks. When ϕ_1 is high and ϕ_2 is low, the input voltage on each branch, i.e., v_i, charges its corresponding capacitor, C_1, in that branch. When ϕ_2 is high and ϕ_1 is low, charges in the branch capacitors are transferred to the integrator capacitor, i.e., C_2 in the feedback path of the op-amp.

If s inputs ($0 \leq s \leq m$) added together, the output voltage is given by (4.2).

$$v_o = -\frac{C_1}{C_2}\Big(v_1 + v_2 + ... + v_s\Big). \tag{4.2}$$

In the case of time variant signals, the addition is performed for following samples of the input signal. Therefore, at end of each addition interval, the capacitor, C_2 should be reset. Also, the analog adder in Figure 4.3 provides negative outputs. Therefore, a positive DC voltage V^+ is applied to the positive input of the amplifier to shift the outputs into the positive voltage range. Figure 4.4 shows the modified integrator structure.

Note that the output voltage right before each rising edge of ϕ_1 is expressed by the branch voltages, i.e., $v_1, v_2, ..., v_s$, which are sampled at the previous rising edge of ϕ_1 and s is the number of inputs which are added together. In our work, s is the number of pixels which are selected by the random sequence to be added together. In other words, for each CS measurement, s is the number of 1's in its corresponding row in the measurement matrix, $\mathbf{\Phi}$.

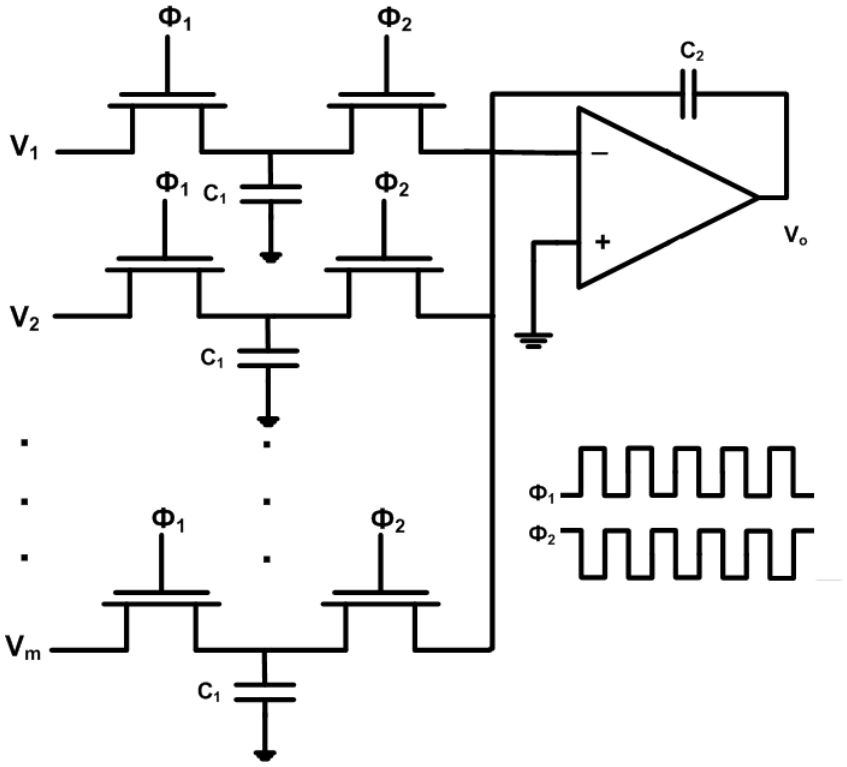


Figure 4.3: General schematic diagram of the analog adder.

As can be seen in Figure 4.3, m SC branches are connected to the negative input of the op-amp. The structure of SC branches should be modified to provide the possibility of selecting s branches out of m for different CS measurements. Figure 4.5 shows the modified structure for the SC branch. One *select* signal is connected to each branch. When the *select* signal is high, transistor M_7 is off and the switching clocks are applied to the gate of transistors M_8 and M_9. Therefore, the SC branch is working as expected and shown in Figure 4.3. When the *select* signal is low, transistor M_9 is off and SC branch is disconnected from the integrator. Also, transistor M_7 is on to suppress possible leakage from the input voltage of the SC branch. The capacitor is also grounded to make sure that there is no charge from

the past when the branch is selected to contribute to measurement values for the next measurement extraction interval.

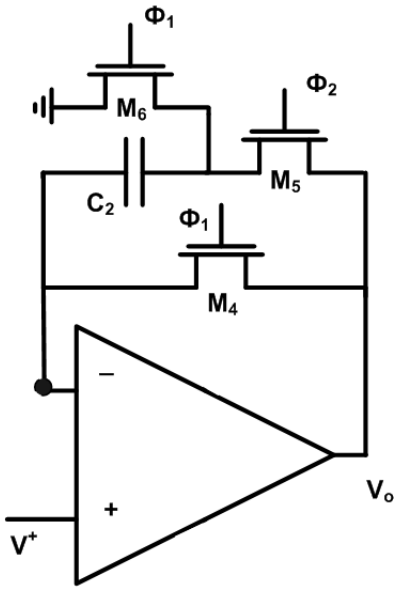


Figure 4.4: Modified integrator structure.

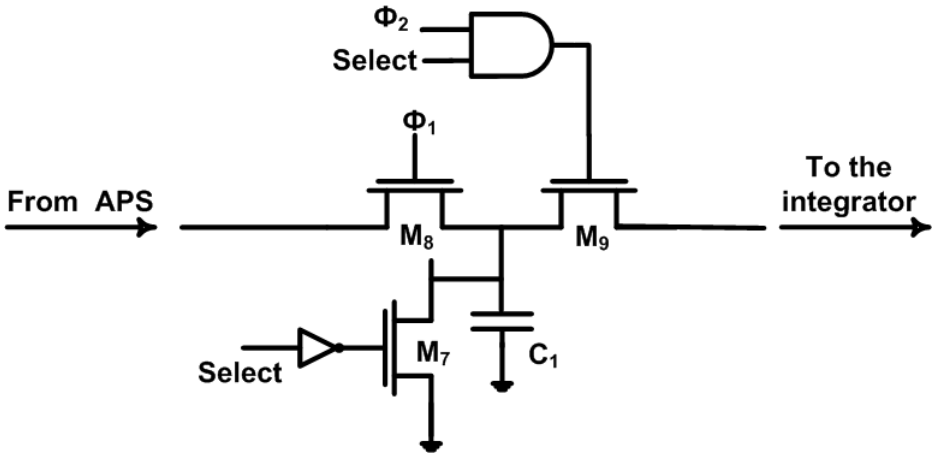


Figure 4.5: Schematic diagram of the switched capacitor (SC) branch.

An analog adder with 4 branches was designed and simulated to test the performance of the analog adder and the selection of different SC branches. Figure 4.6 shows the simulation results for four different *select* sequences. The input voltage of each branch is $500mV$ and $V^+ = 700mV$. It can be seen that by selecting more branches the output voltage increases monotonically during the integration of charge in C_2.

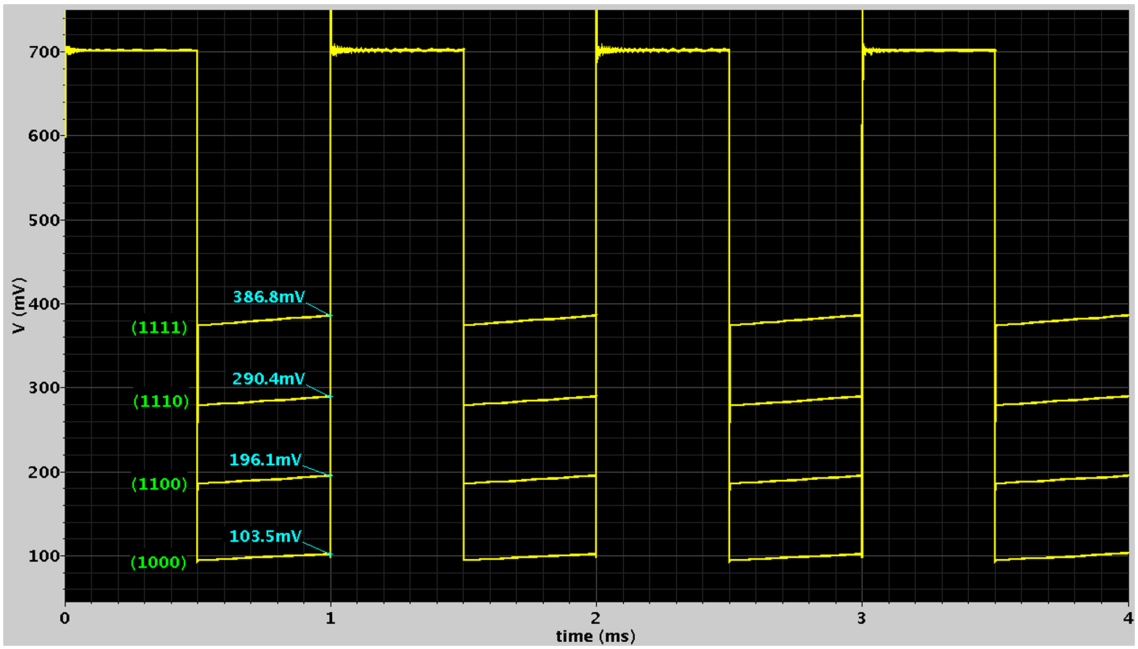


Figure 4.6: Output of the analog adder with four SC branches for different *select* sequences.

4.3 Voltage-mode CS sensor

The APS structure and analog adder circuit should be connected together to implement the CS encoding. One of the difficulties in the design is adding the required circuits to the imager to form the CS measurements while avoiding extra in-pixel elements. The other

challenge is the read-out strategy and the method of implementing the measurement circuitry in such a way as to make the design scalable for any array size.
In this section, our method of making the measurements and comments on the implementation of the read-out and interconnections in the array for the design of the imager, will be described.

4.3.1 Measurements technique

The analog adder and the APS structure which were mentioned in the previous section can be exploited to develop the CS encoding for a CMOS imager. Based on (4.1), since the output voltage of each pixel is related to the photocurrent of the pixel, then by adding the pixel outputs together, the aggregate level of the light can be obtained. The SC branches in Figure 4.3 can be connected to the output of APS pixels to make the CS measurements.
By combining equations (4.1) and (4.2), the output of the analog adder in which its inputs are connected to the output of s randomly selected APS pixels is given by:

$$v_o(t) = \frac{C_1}{C_2}\Big(\frac{t}{C_{PH}}(i_{PH_1} + i_{PH_2} + ... + i_{PH_s}) - sv_{o_{RS}}\Big). \tag{4.3}$$

For the CS encoding, since each measurement represents a summation of the intensities of the selected pixels, which is the aggregate incoming light, then there should be a linear relation with constant coefficients between the aggregate photocurrent and the measurement value. Since different random sequences have different number of 1's, then the parameter s in (4.3) varies for different measurements. Therefore, there is a DC shift i.e., $sv_{o_{RS}}$, in the output voltage, which is independent of the light intensity and it varies for different measurements.
To make an appropriate and universal interpretation from (4.3) and to compensate for the

effect of the term $sv_{o_{RS}}$, each $v_o(t)$ could be sampled twice, i.e., correlated double sampling (CDS), [75]. Sampling two times results in:

$$v_o(t_2) - v_o(t_1) = \frac{C_1(t_1 - t_2)}{C_2 C_{PH}}\Big(i_{PH_1} + i_{PH_2} + ... + i_{PH_s}\Big), \tag{4.4}$$

where t_1 and t_2 could be the first possible time after the reset and the last possible time before the next reset, respectively. Therefore, by choosing the same sampling times, t_1 and t_2, for all measurements, the voltage in (4.4) can be interpreted as the CS measurement.
As mentioned before, (4.2) is valid for the rising edge of ϕ_1, i.e., the falling edge of ϕ_2. Therefore, the discrete time equivalent of (4.3) and (4.4) could be rewritten as follows.

$$\begin{aligned} v_o(n) &= -\tfrac{C_1}{C_2}\Big(v_1(n-1) + v_2(n-1) + ... + v_s(n-1)\Big) \\ &= \tfrac{C_1}{C_2}\Big(\tfrac{(n-1)T}{C_{PH}}(i_{PH_1} + i_{PH_2} + ... + i_{PH_s}) - sv_{o_{RS}}\Big), \end{aligned} \tag{4.5}$$

$$v_o(n_2) - v_o(n_1) = \frac{C_1(n_2 - n_1)T}{C_2 C_{PH}}\Big(i_{PH_1} + i_{PH_2} + ... + i_{PH_s}\Big), \tag{4.6}$$

where T is the period of the sampling clock, ϕ_1, and $v_o(n)$ is the output which is sampled at n^{th} rising edge of ϕ_1.
Note that the output voltage of each APS pixel in (4.1) is always positive. Therefore, during the integration time, $v_{o_{RS}} \geq \frac{i_{PH}}{C_{PH}}t$. It results in a negative value for the voltage in (4.3). However, considering the DC offset, V^+, which is applied to the positive input of the op-amp, the output voltage can be rewritten as follows.

$$v_o(t) = \frac{C_1}{C_2}(v_{d_1} + v_{d_1} + ... + v_{d_s}), \tag{4.7}$$

in which $v_{d_i} = (V^+ - v_i)$.

Therefore, if $V^+ > v_{o_{RS}}$ and $\frac{C_1}{C_2} < \frac{1}{s_{max}}$, then the output is always positive and less than V^+. As discussed in the next sections, in our design, the CS-measurement is performed for 4×4 blocks by using 16 SC branches. Therefore, $s_{max} = 16$ and $\frac{C_1}{C_2}$ should be less than $\frac{1}{16}$.

Figure 4.7 shows the simulation results for four APS pixels connected to the integrator via four SC branches. The photocurrent of $1pA$ is considered for all pixels and $V^+ = 650mV$. The results show four different states when 1, 2, 3, or 4 pixels are selected to contribute in the CS measurement integration. The 10 following samples of the output are shown for each state. It can be seen that the output increases more by adding more pixels together. Also, the envelope of following samples for the output in each state is proportional to the output of the APS pixels (Figure 4.2).

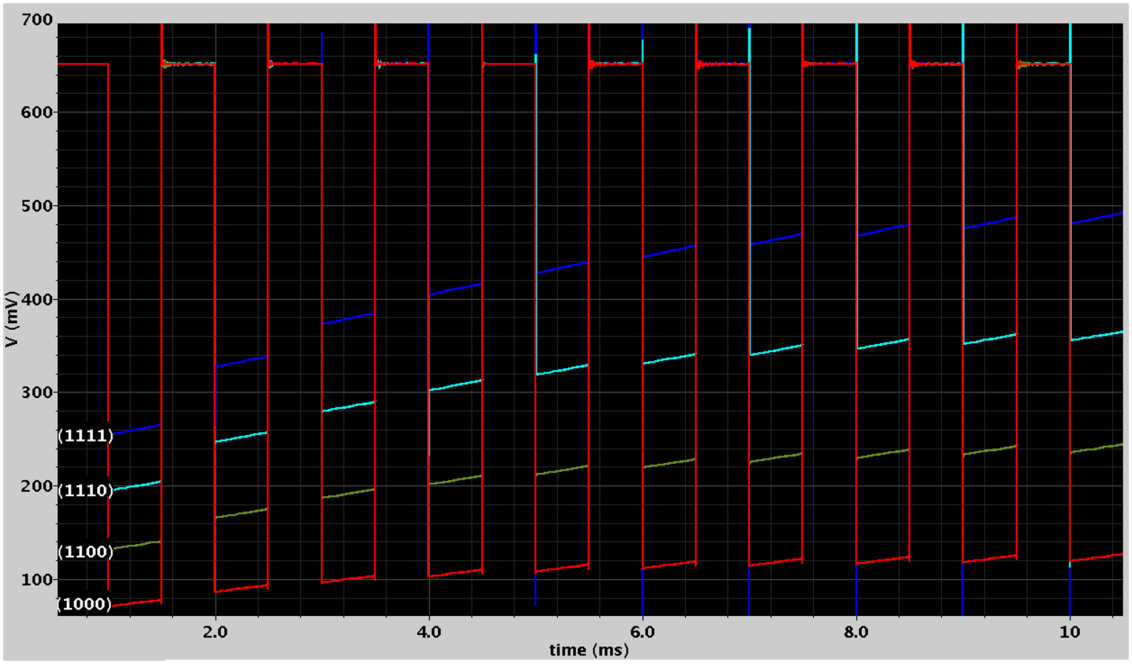


Figure 4.7: Output of the CS-CMOS for four pixels connected to the analog adder and different *select* sequences.

4.3.2 Block coding

For each measurement, only some of the pixels are added together, so there should be a proper strategy for selecting the pixels based on the measurement matrix. Also, the read-out and the interconnections of the array, e.g., connections between the APS pixels and SC branches, should be performed in such a way that the fill-factor of the imager remains similar to the imager with no compression. Moreover, the design should be scalable for any desired array size.

Each pixel needs to be connected to an SC branch and have a select signal to make it effective in some measurement values when its corresponding element in the measurement matrix is 1. Note that the select signals are connected to the SC branches instead of the pixels. The SC branches could be implemented inside the pixel, but in this case, the fill-factor decreases and one select signal should be directly connected to each pixel via one individual signal path. Also, the SC branch could be implemented outside the array to improve the fill-factor, although an individual signal path is still required to connect the output of the APS to its corresponding SC branch.

To have a high fill-factor, we moved the SC branches outside the array. Since each pixel needs a separate signal path toward its SC branch which is outside the array, more separate signal paths are required for each column when the size of the array increases. As mentioned in the previous chapter, these metal paths should not cover the photosensitive area of the sensor. Moreover, there is a limited number of available metal layers in each technology. By increasing the number of pixels in the array, the number of parallel metal paths within columns will increase and the spatial resolution of the imager will degrade. It is not practical to implement this design for the imagers with a large array size. Therefore, the design is implemented for separate blocks because of the layout restrictions and other

limitation which is mentioned in chapter 3, section 3.2.3, i.e., LFSR size, scalability, and reconstruction time. Figure 4.8 shows a 4×4 block of APS pixels and their parallel paths toward the SC branches, four paths for each column.

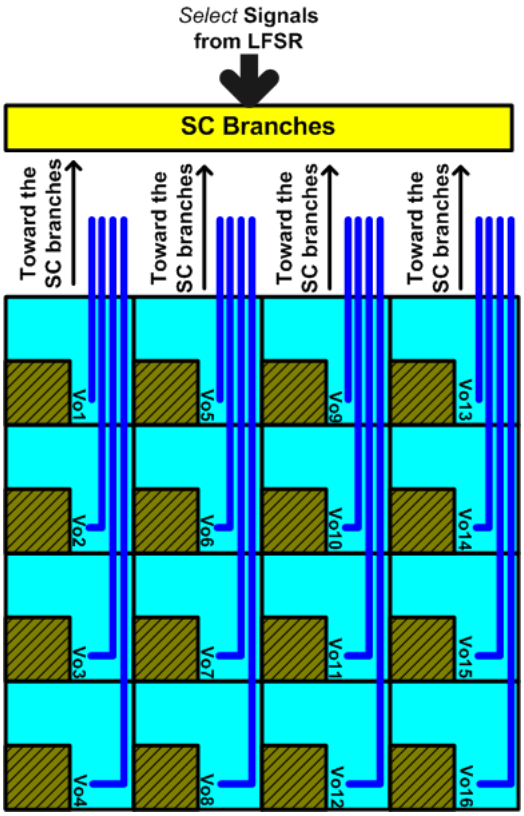


Figure 4.8: Read-out structure for one 4×4 block.

To make an array with any desired size, these blocks could be stitched together as shown in Figure 4.9 (a). The 16×16 array consists of 16 blocks, 4 rows and 4 columns of blocks. The SC circuit has 16 branches corresponding to the 16 pixels in each block. Note that SC branch are shared among all blocks. Therefore, parallel paths of the blocks are connected together and 16 SC branches are connected to them. For example, the pixels in the first row

and first column of each block are connected together and they share one SC branch. The read-out process is performed for the blocks one by one. This is the same as block-by-block (BB) read-out scheme mentioned in the previous chapter.

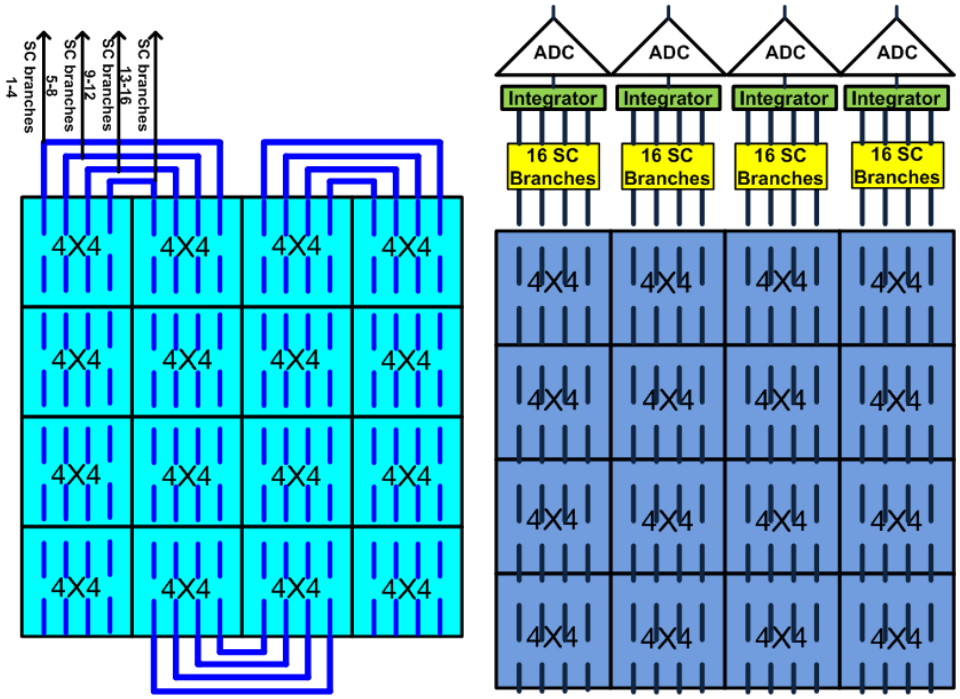


Figure 4.9: (a) Connection of 16 blocks to make a 16×16 array (block-by-block (BB) read-out), (b) Connection of 16 blocks to make a 16×16 array (column-of-blocks (CB) read-out).

The column-of-blocks (CB) structure can be used to improve the read-out time of the array (see equations (3.5) and (3.6)). Figure 4.9 (b) shows the structure for the CB read-out scheme with 16 SC branches, one integrator and one ADC for each column of blocks. Figure 4.10 shows the schematic diagram for the CS sensor circuitry. The diagram shows only one pixel. Other pixels could be connected to the negative input of the operational amplifier via their own SC branches, i.e., the middle block in the Figure 4.10.

It is worth mentioning that the integrator part is off the array and is shared among several blocks. Also, the SC branches are off the array and each of them is used in common among all corresponding pixels in several blocks.

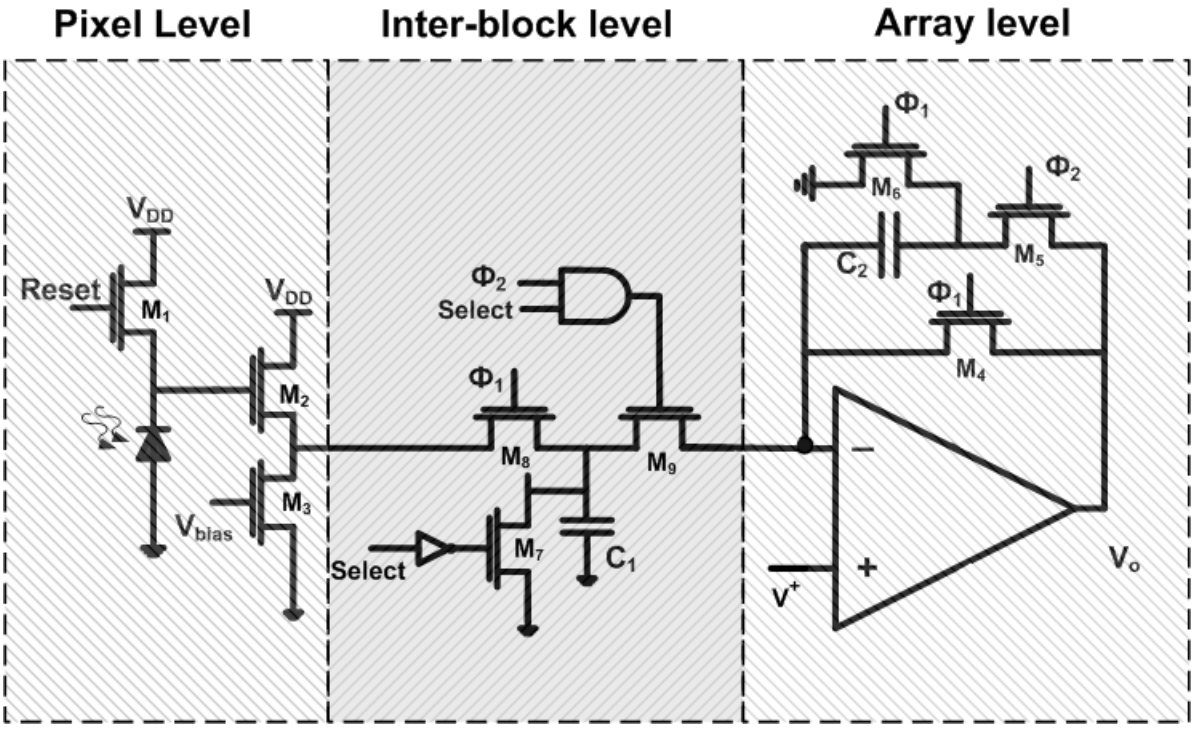


Figure 4.10: Schematic diagram of each pixel connected to the integrator via its SC branch.

4.3.3 Switching errors

There are two sources of switching noise which can affect the performance of the charge transfer from the pixel to the sampling capacitor, C_1. Channel charge injection and clock feedthrough of the switching transistor, M_8 in Figure 4.10, can create an error in the amount of the transferred charge.

Channel charge injection

Ideally, there should be no current injection through the switching transistor during its off period. However, the accumulated charge in the inversion layer of the transistor moves toward the source and the drain. The charge injected to the sampling capacitor can change the value of the sample extracted in the previous period. As this value is affecting the output of the integrator, the injected charge creates an error in the CS measurement value. The amount of the charge injected toward the sampling capacitor can be expressed as follows,

$$Q_{ch} = WLC_{ox}(V_h - V_{op} - V_{th}), \tag{4.8}$$

where C_{ox} is the oxide capacitance per unit area, V_h is the level of the voltage for the clock in its high state, V_{op} is the pixel output value and V_{th} is the threshold voltage of the transistor. If half of this charge is injected toward C_1, the error in the sample value is:

$$\triangle V = \frac{WLC_{ox}(V_h - V_{op} - V_{th})}{2C_1}. \tag{4.9}$$

Therefore, the amount of the error for the measurement value, i.e., output of the integrator, is:

$$\triangle V_m = \frac{C_1}{C_2}\triangle V = \frac{WLC_{ox}(V_h - V_{op} - V_{th})}{2C_2}. \tag{4.10}$$

From (4.10), it is noted that the amount of the error decreases by increasing C_2 compared to WLC_{ox}.

Clock feedthrough

The clock applied to the gate of the switching transistor is coupled to the sampling capacitor through the gate-source overlap capacitance. The amount of error is calculated as follows,

$$\triangle V_f = V_h \frac{C_{gso}}{C_{gso} + C_1}, \tag{4.11}$$

where C_{gso} is the overlap capacitor between gate and source. Also, the amount of the error for the measurement value $\triangle V_{fm}$, is:

$$\triangle V_{fm} = \frac{C_1}{C_2} \triangle V_f = V_h \frac{C_{gso}}{C_2(1 + \frac{C_{gso}}{C1})}. \tag{4.12}$$

Therefore, by choosing large values for C_1 and C_2 compared to the gate-source capacitance, the effect of the clock feedthrough can be mitigated.

Although choosing large values for C_1 and C_2 decreases the switching errors, it can also decrease the speed of the sampling and integration steps. In the sampling period, when transistor M_8 is on, C_1 is being charged with the time constant equal to $R_{on8}C_1$, where R_{on8} is the source-drain resistance when the transistor is on. This time constant should be small enough to complete the charging of C_1 before the next integration period. Also, C_2 is integrating the charge in different branches with the time constant $R_{on5}C_1$, where R_{on5} is the source-drain resistance for transistor M_5. This time constant should also be small enough to complete the integration before the next sampling interval. Therefore, the trade-off between the speed of charging and the switching noise has been considered to find appropriate values for C_1 and C_2. In this work, we used $C_1 = 200fF$ and $C_2 = 3.2pF$ to mitigate the switching noise and also fulfill the ratio of $\frac{1}{16}$ mentioned in section 4.3.1.

Considering $WLC_{ox} = 4.2fF$, $V_h = 1.2V$, $V_{th} \simeq 500mV$ and $C_2 = 3.2pF$, the maximum charge injection error is:

$$(\triangle V_m)_{max} = \triangle V_m|_{V_{op}=0} = 0.45mV. \tag{4.13}$$

Also, $C_{gs} = 1.6fF$ and $C_1 = 200fF$. Therefore,

$$\triangle V_{fm} = 0.59mV. \tag{4.14}$$

The source-drain on-resistance for the switching transistors are about 1kΩ which leads to the time constant of $0.2nsec$. Therefore, the charge transfer can be completely performed for the clock with frequencies up to $1GHz$.

The effect of switching errors can be measured by turning off the reset signal and measuring the output. The outputs of the pixels, i.e., the inputs of SC branches, are zero while the reset signal is off. Therefore, using (4.7), the expected output is:

$$v_o(t) = s\frac{C_1}{C_2}V^+, \tag{4.15}$$

where s is the number of ones in the selecting sequence. The difference between the measured and expected output is the effect of the switching errors. In our measurements, a 4×4 block and selecting sequence with 7, 8, and 9 ones have been used. With $V^+ = 600mV$, the expected values for the output were $262.5mV$, $300mV$ and $337.5mV$ for 7, 8, and 9 ones, respectively. Measurement results show deviations of $6.9mV$, $8mV$, and $9.2mV$ from the expected values. It shows the switching error of around $1mV$ for each active SC branch. Note that, this error will be reflected in the noise upper bound in the reconstruction problem.

4.4 Experimental results and discussions

4.4.1 16×16 array

Using 4×4 blocks and the BB read-out structure, a 16×16 array was designed to investigate if the idea is scalable to any desired array size. Figure 4.11 shows the layout for a 16×16 array. Here, 16 blocks are connected together and there are 16 SC branches at the top and

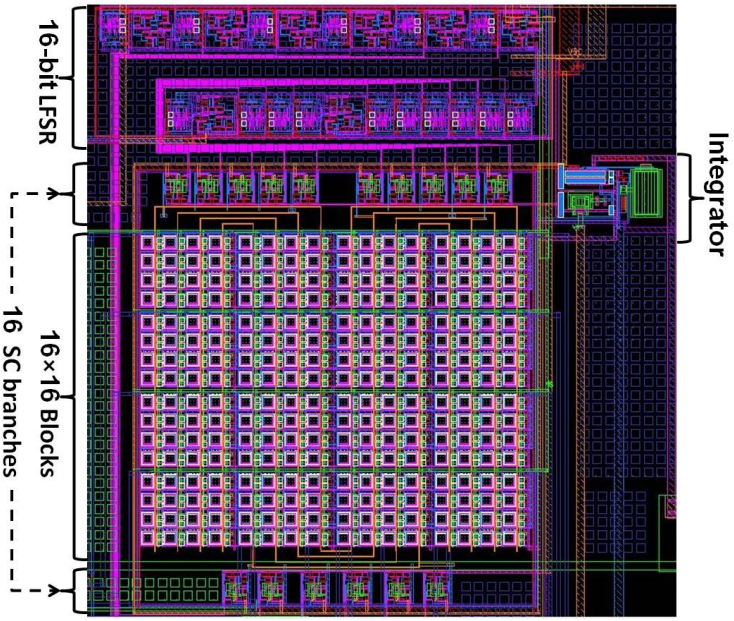


Figure 4.11: The layout for a 16×16 imager

bottom of the array. A 16-stage LFSR circuit, at the top of the figure, is connected to the SC branches. As can be seen in the figure, all blocks are connected together, similar to the blocks in Figure 4.9 (b). Using the same method of connection, imagers with larger sizes can be built by 4×4 blocks. If the measurement process works for any 4×4 block and the block by block read-out works for all 16 blocks, the feasibility of the design for larger array

sizes is corroborated.

4.4.2 2×2 block

To verify the functionality of the whole design as a valid CS encoder and to investigate the accuracy and monotonicity of the measurement values, a 2×2 block was designed and fabricated in 130 nm CMOS technology. The selecting signals, e.g., values of the measurement matrix, for this array are set from outside the chip to investigate the effect of adding different number of pixels to the output.

In our measurement process, 16 sampling periods of the adder are considered during the integration time of the pixels, as shown in Figure 4.12. Therefore, there are 16 available samples of the output voltage as a discrete version of (4.3), i.e., (4.5).

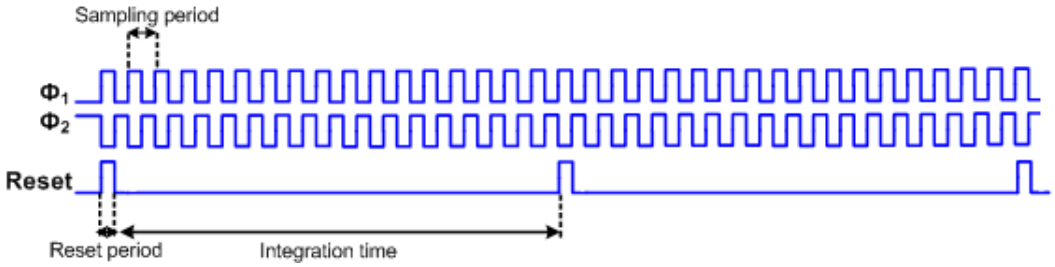


Figure 4.12: Timing diagram for the reset and switched capacitor clocks.

Although the output of the APS cells are valid during the whole integration time, the output of the adder is valid only when ϕ_1 is low and ϕ_2 is high, i.e., the charging period of C_2. The functionality of the adder is based on the two separate steps, i.e., sampling and adding. During the sampling period, the outputs of the selected pixels are integrated in their own branch capacitors and the output of the array is set to the voltage equal to the positive input

of the Op-Amp. Therefore, regardless of the level of the light or the number of effective pixels in the measurement, the output of the adder is constant during the sampling interval. During the adding period, the aggregate charge of the selected pixels are integrated in C_2. Also, as expected from (4.5), the trend of the output is increasing for consecutive samples between the two following reset periods. Since selecting more pixels leads to more aggregate light, i.e., slope of the curve in (4.3), then the highest rate of increase is for the case when the output of all pixels are selected and added together. Considering the linearity assumption in (4.1) and as can be interpreted from (4.3) and (4.4), the differential output, or the slope of the output, is proportional to the aggregate level of the light intensity.
Figure 4.13 shows the output of the adder between two following reset periods for different selecting sequences and at same level of light for all four pixels. As the number of selected pixels increases, the slope of the curves increases proportionally. Therefore, the maximum rate of increase is in Figure 4.13 (d) where the outputs of four pixels were added together and minimum rate is for Figure 4.13 (a) which shows the output of the integrator when only one pixel is effective in the addition.

To investigate the trend of the output samples, Figure 4.14 shows the amount of voltage for 16 samples for different numbers of selected pixels. The value of each sample mentioned on the curve is the final aggregated charge stored in C_2 for that sample. It can be seen in Figure 4.14 that for each sample, there is a monotonic increase in the output voltage while the number of the selected pixels is increasing. Based on the linear fitting of the curves, the slopes of the curves are 1.5, 3.2, 4.4 and 6.4 $mV/sample$ for selecting 1, 2, 3, or 4 pixels, respectively. The slopes are almost proportional with some errors because of the device mismatch in different SC branches. Since the slope of the curves represents the aggregated photocurrent, which is the CS measurement value, then having proportional

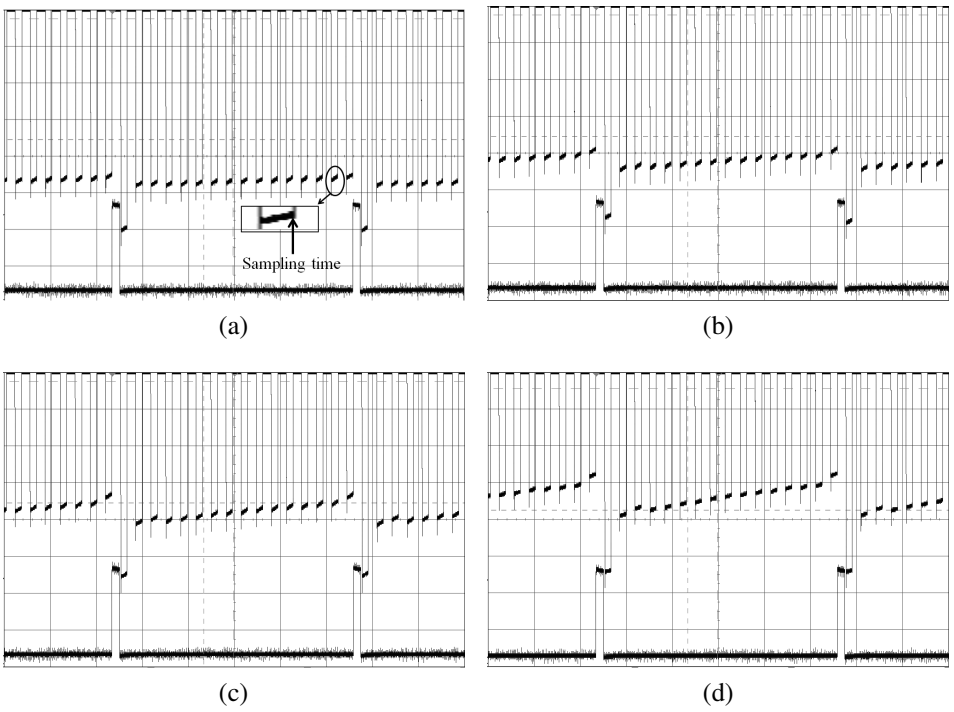


Figure 4.13: Resulted output for a 2×2 block.(a) One selected pixel, (b) Two selected pixels, (c) Three selected pixels, (d) Four selected pixels.

slopes confirms the validity of CS encoding process.

4.4.3 4×4 block

The design was implemented for a 4×4 block. Pixels in the 4×4 block are selected using the random sequences produced by LFSR, i.e., *select* sequences. Therefore, the number of pixels which are added for each measurement equals the number of ones in its corresponding *select* sequence coming from LFSR. Figure 4.15 shows the trend of number of ones and measurement values for different *select* sequences and four levels of light with uniform illumination on the entire block. The difference between the second and second-to-last samples between two following reset signals has been considered as the measurement value. As can

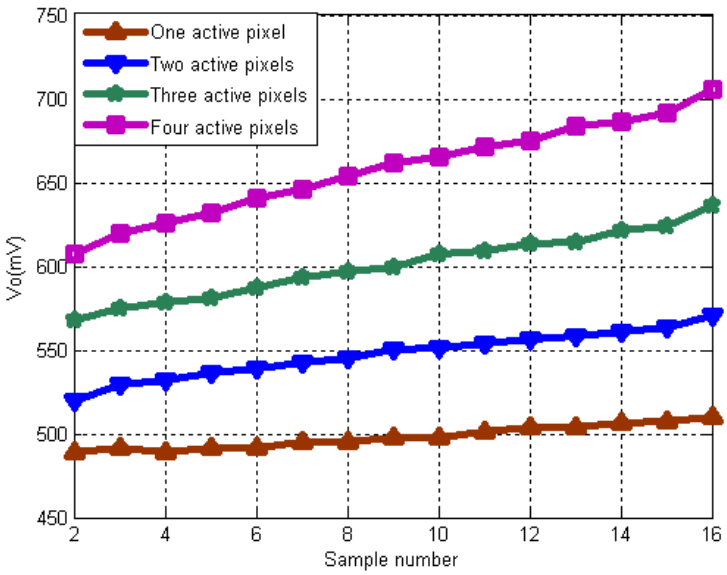
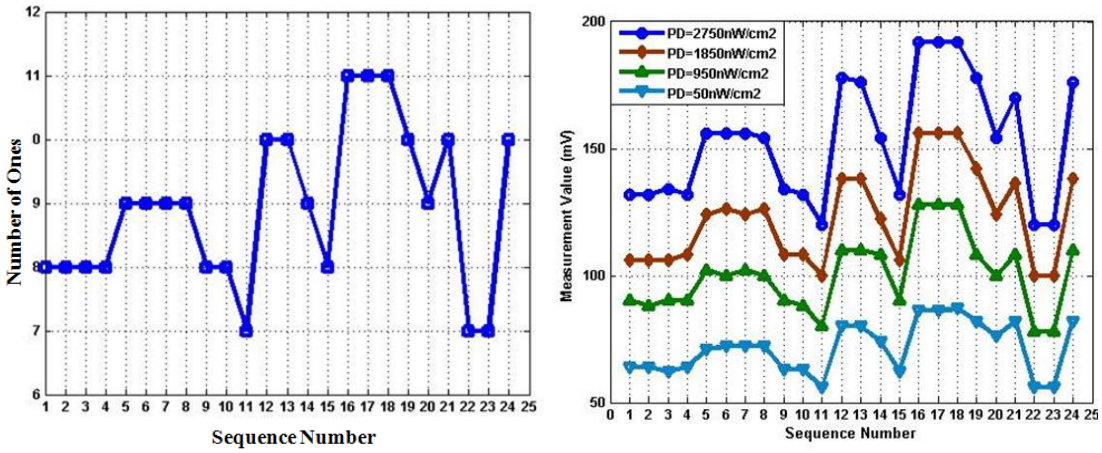


Figure 4.14: Consecutive samples of the analog adder output for 2×2 block and different number of selected pixels.

be seen in the figure, the trend of the measurement values is consistent with the trend of the number of ones. This result confirms the validity of the selecting process using the LFSR output for the 4×4 block which is the building block of a sensor with any desired array size.

Figure 4.16 shows the oscilloscope's screenshot for the output of the 4×4 block, reset signal and LFSR *clear* signal. When the *clear* is high, none of the pixels are selected for the measurement. When the *clear* goes high, LFSR starts working to produce different random sequences consecutively. The amount of the rise for each sample is proportional to the number of ones in the *select* sequence based on the Figure 4.15(a).



(a) (b)

Figure 4.15: (a) Number of ones for 20 following *select* sequences. (b) Measurement values for 20 following *select* sequences and different levels of light.

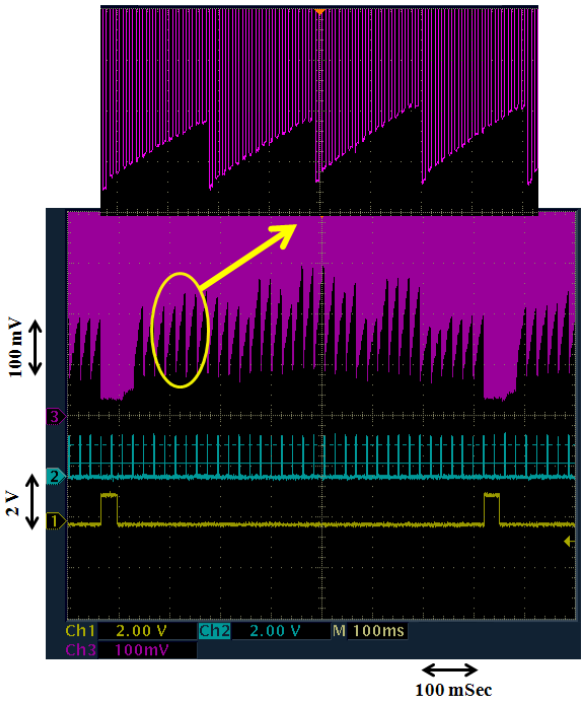


Figure 4.16: Output for a 4×4 block (channel 3), reset signal (channel 2), and *LFSR clear* signal (channel 1).

Figure 4.17 shows the measurement values as a function of the level of the uniform light illumination for three different *select* sequences with 7, 9 and 11 number of ones. The dashed lines show the linear fit for each set of measurements. The linear trend of the curves confirms the linear relation of the measurement values and the level of the light, as expected from (4.6). Also the slopes are 0.023, 0.03, and 0.036 for 7, 9 and 11 selected pixels, respectively. This shows the proportionality of the measurement values with number of the selected pixels.

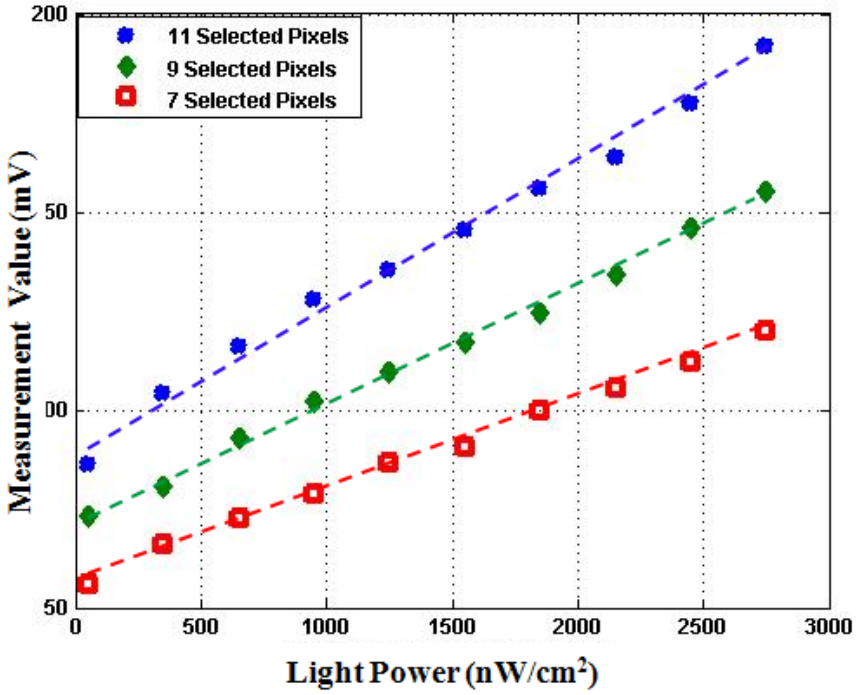


Figure 4.17: Measurement values for different levels of light, 4×4 block and different number of selected pixels.

Figure 4.18 show the measurement values with respect to number of selected measurements. As shown in the figure, the measurement value for zero selected signal is very small

for all levels of light. It means that non-selected pixels have no effect on the measurement values. It is because of the structure of the SC branch (see Figure 4.5). Transistor M_7 and AND gate prevent the APS pixel from affecting the measurement value when the *select* signal is zero. However, the switching transistors in the integrator structure (see Figure 4.4) might affect the output voltage even when all of the branch are disconnected from the integrator. But, because of the correlated double sampling (CDS) (see (4.4)) the effect of this error is not noticeable.

Figure 4.19 shows the SNR values for different levels of the light and different number

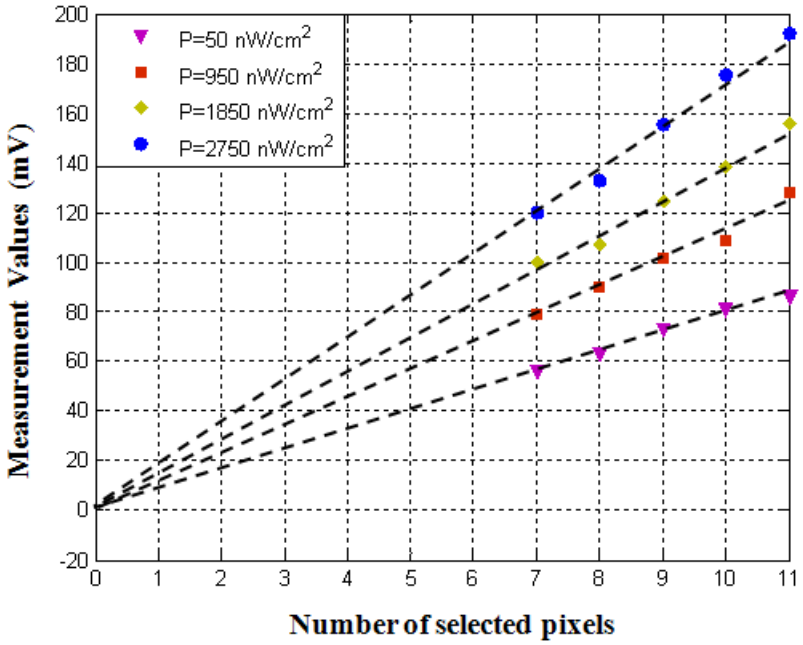


Figure 4.18: Measurement values for different number of selected pixels, 4×4 block and 4 different levels of light.

of selected pixels for the CS measurements. For each curve and for each level of the light, 50-100 different samples of the output were measured. By considering the mean value of

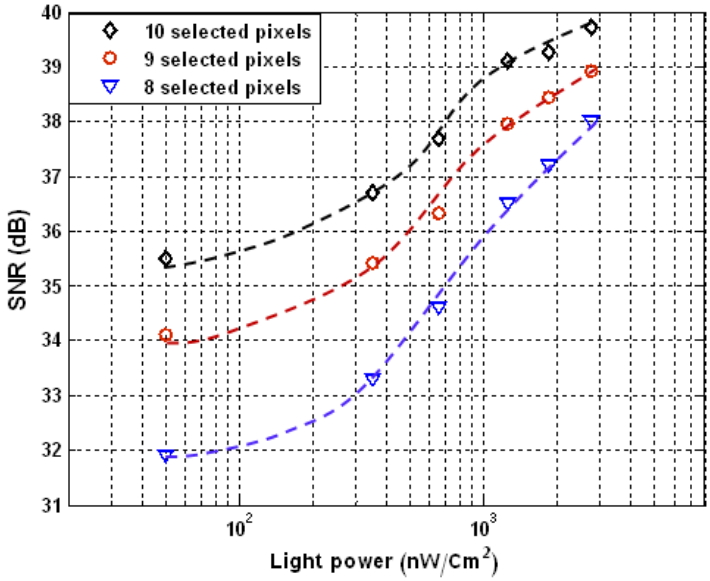


Figure 4.19: Signal-to-noise ratio (SNR) for different levels of the light and different number of selected pixels.

the samples as the signal and the variance as the noise, the SNR has been calculated. As can be seen in the figure, by increasing the number of selected pixels and level of the light, the amount of the SNR increases. It is because of the dependency of the signal power to both level of the light and the number of selected pixels.

Using the SNR values extracted from the SNR graph, the effect of the noise in image reconstruction can be investigated. Figure 4.20 shows the simulation results for the Cameraman image and different number of noisy measurements. The same simulation has been presented in Figure 3.22 for the current-mode design. By comparing the PSNR values, it can be seen that the higher SNR values in voltage-mode design lead to a higher image reconstruction quality. However, this improvement is more noticeable for higher number of measurements (see the PSNR values for M=12) where the reconstruction error is less dominant compared to the noise in CS measurement values.

(a) PSNR=26.5 dB (b) PSNR=22.31 dB (c) PSNR=19.9 dB

Figure 4.20: Cameraman image and its reconstructed versions for different number of noisy measurements, (a) M=12, b) M=8, and (c) M=4.

Chapter 5

Multiple-camera CS video coding

5.1 Introduction

A video signal consists of a sequence of correlated image frames. Considering each frame as an array of pixel values, then a large amount of data should be sent off the camera to represent the captured video. Therefore, video compression algorithms are used to reduce the required bandwidth for communications or storage for the video signal. Also, for the same bandwidth or storage, a higher frame rate is achievable by using video compression. The video compression algorithms use the dependency of the pixels on spatial domain as well as the redundancy of signal on temporal domain [21]. The temporal redundancy depends on the amount of the motion in different spots of the scene, and the motion information of the scene is used to represent following video frames with less amount of data.

The amount of the data depends on the size of the array, number of the bits per pixels and the frame rate. The higher the frame rate, the higher is the amount of the data for a specific time interval of the video signal. In video recording of the scenes with fast motions and sudden movements, a very high frame rate acquisition is required. Therefore, efficient video compression methods are necessary and helpful for high-speed applications. Some

of the applications of high-speed cameras are in auto crash testing [103], experimental fluid mechanics [104], high-speed video microscopy [105] and scientific study of welding process [106], explosions [107] and dynamics of transient phenomena [108].
In this chapter, the problem of CS video coding is studied and we present a sensor structure which is suitable for the multiple-camera CS video coding architecture.

5.2 CS-CMOS video coding

The CS-CMOS sensors designed for image acquisition can also be used for video coding by encoding each frame of the image separately. In this case, the measurement values for each frame should be captured and sent off the camera before starting the measurement process for the next frame. The entire video acquisition scheme for n consecutive frames of the video is shown in Figure 5.1.

Figure 5.1: Timing diagram for frame-by-frame video coding using consecutive image coding steps.

In the CS image decoding process, the spatial dependency of different pixels in the image is exploited as the sparsity objective function. However, there are temporal correlations between the consecutive frames of a video which can be used to improve the reconstruction performance of a CS-encoded video.
In [109], the temporal information of the video was exploited to improve the overall reconstruction time. In one implementation, the decoded image for each reconstructed frame

was used as the initial value for the reconstruction problem for the next frame. In another implementation, the CS decoding problem was solved for multiple frames at the same time with reasonable computation time by using the similarities between subsequent frames. The sparsity of the image derivative in temporal domain was used in [109]. However, the motion function of the video from one frame to another can also be used to improve the reconstruction problem. In [110], considering the relationship between motion and CS coding, a method to estimate the motion information of the image was presented. Also, the motion parameters were included in the reconstruction problem to improve the quality of decoded frames. However, the encoding structure shown in Figure 5.1 was used for the measurement extraction.

Considering the timing diagram in Figure 5.1, the video frames cannot be captured during the read-out intervals. The fact that the read-out time increases with increasing the size of the imager restricts the frame rate of the CS video camera. However, the measurement process can be performed in temporal domain as well as the spatial domain to improve the capturing time of a constant number of frames. Therefore, the measurements can be captured for a three-dimensional cube of data, Figure 5.2 (a), instead of separate measurements for several two-dimensional data sets, i.e., image frames. As shown in Figure 5.2 (b), there is only one read-out interval by extracting the measurements for a cube of n following frames. The resulted measurement values can be used to recover the video data for all n frames. By reducing the number of read-out intervals the frame rate of the system increases.

In [111], a new multiple-camera CS video coding based on the read-out scenario shown in Figure 5.2 (b) is presented. The measurement process is performed in time domain using multiple cameras for the same scene. Spatial and temporal correlations in video were used

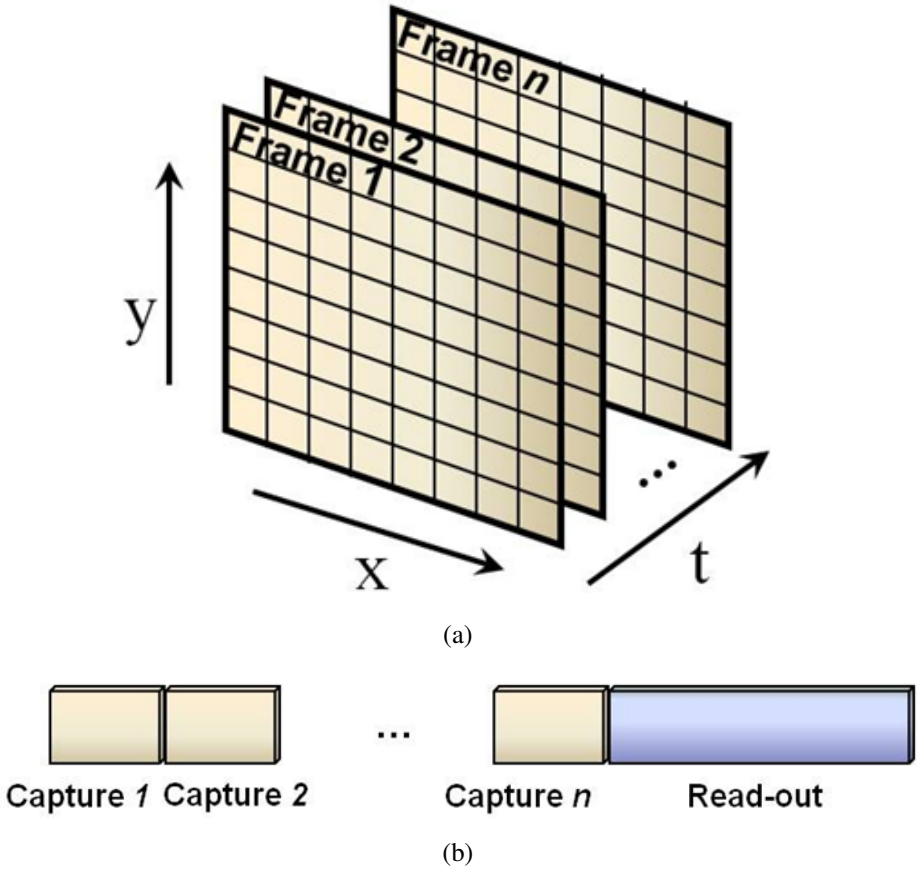


Figure 5.2: (a) Cube of data for n frames, (b) Multiple-capture and one read-out timing diagram.

to define the reconstruction problem. In the next sections, the measurements method and the hardware implementation for the mentioned algorithm are studied.

5.3 Multiple-camera CS measurements

The algorithm presented in [111] can be implemented using frame-level (FL) or pixel-level (PL) approach. In the FL approach, an n-bit random binary sequence is assigned to each camera. The random sequences correspond to the n following frames of the video. The level of the light for all pixels in the frame is integrated if its corresponding bit in the

sequence is '1'. The light is accumulatively integrated for different frames to produce the CS measurements. The random measurement process is completely performed in temporal domain. Since all pixels in each frame have the same random bit, then the measurement process is not random in spatial domain.

Considering a video segment including n following frames of the video as matrix $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_n\}$, then the CS measurement values for k^{th} camera in FL implementation is:

$$\mathbf{y}_k(\text{FL}) = \sum_{i=1}^{n} a_{i,k}\mathbf{f}_i, \tag{5.1}$$

where $a_{i,k}$ is the random coefficients for i^{th} frame ($\mathbf{f}_i$) in k^{th} camera.

To have randomness in both spatial and temporal domains, each pixel can have a separate random sequence. In the pixel-level (PL) approach, one binary random vector with n entries is assigned to each pixel of the image in each camera. If the i^{th} bit of the vector is '1', the level of the light for that pixel in i^{th} frame is integrated and added up to the measurement value for that pixel. The level of the light during each frame does not affect the measurement value when its corresponding bit in the random vector is '0'. Figure 5.3 shows the pixel selection for the temporal CS measurement process and also the expected response of one pixel for $n = 8$ and measurement vector of "01011101".

One measurement value is extracted for each pixel at the end of the integration time, i.e., the time between two following reset periods. Therefore, the output of each camera is a 2 dimensional matrix of measurement values for each n following frames. The CS measurement values for k^{th} camera in PL implementation is:

$$\mathbf{y}_k(\text{PL}) = \sum_{i=1}^{n} \mathbf{A}_{i,k} \odot \mathbf{f}_i, \tag{5.2}$$

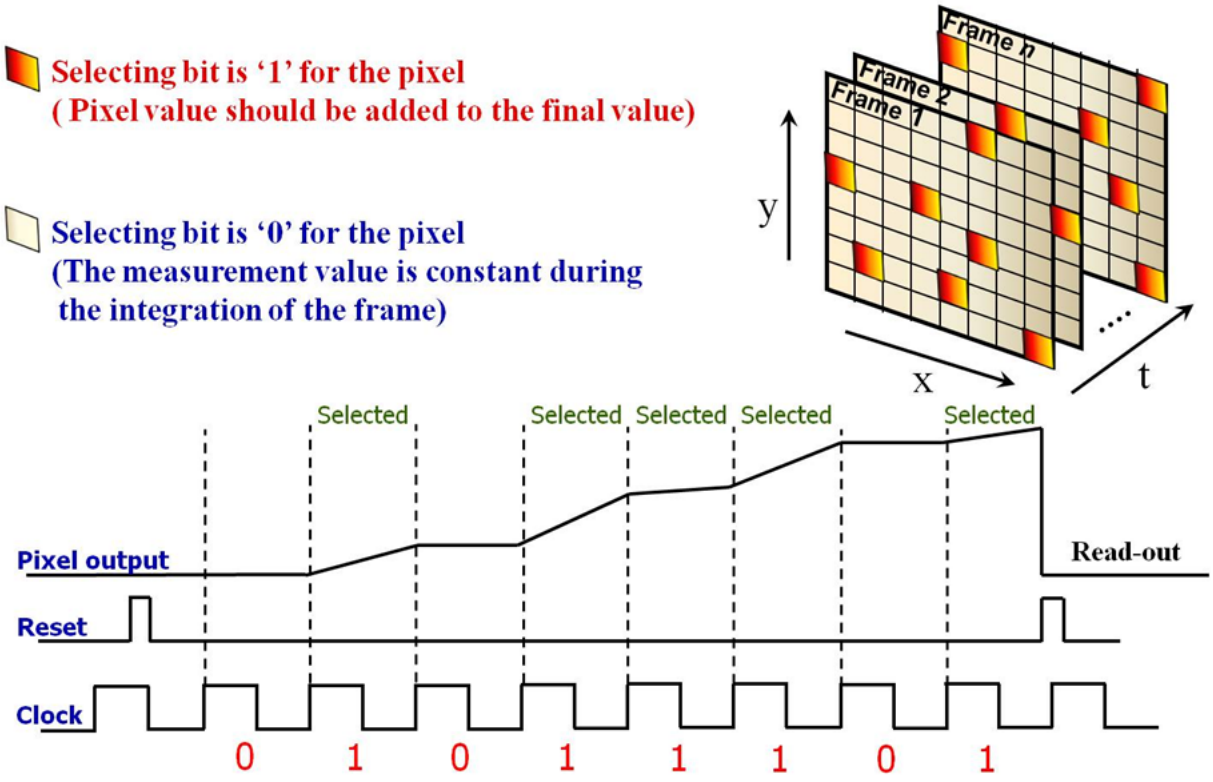


Figure 5.3: Pixel-based, temporal CS measurement for multiple-camera video coding.

where $\mathbf{A}_{i,k}$ is a matrix of random coefficients for different pixels of the i^{th} frame for the k^{th} camera, and $\odot$ is the component-wise multiplication.

For both FL and PL implementations, by concatenating the matrices of CS measurement values captured from different cameras in matrix $\mathbf{Y}$, and inserting the random coefficients in matrix $\mathbf{B}$, the CS measurement process for the entire video coding system can be expressed as:

$$\mathbf{y} = \mathbf{B}\mathbf{f} + \boldsymbol{n}, \tag{5.3}$$

where $\boldsymbol{n}$ is the additive noise for the extracted measurement values.

The reconstruction problem uses the sparsity of the derivative of the video frames in spatial and temporal domains at the same time. The reconstruction problem can be expressed as [111]:

$$\begin{aligned} &Minimize_{\mathbf{f}} \quad (\|\nabla^2_{u,v}\mathbf{f}\|_1 + \|\nabla_{u,v}(\nabla_t \mathbf{f})\|_1) \\ &sub\,ject\ to \quad \|\mathbf{y} - \mathbf{B}\mathbf{f}\|_2 < \epsilon \end{aligned} \tag{5.4}$$

where ϵ is proportional to the power of the measurement noise. Also $\nabla_{u,v}$ and ∇_t are the derivative functions in spatial and temporal domains, respectively.

The PL approach leads to a superior reconstruction quality in the cost of higher reconstruction time. The simulation results for the method is promising and lead to a high frame rate video recording system [111]. However, regular sensor structures cannot be used for the hardware implementation of this design. In the next section, a new hardware implementation suitable for the mentioned algorithm [111] is presented.

5.4 Sensor structure

Conventional CS-CMOS architecture cannot be used as they apply the measurement process along the x-y directions in spatial domain. The pixel design should be capable of adding the integrated light for different frames and keeping the value until the read-out time. Also, the random bit for each pixel changes over time. Therefore, individual access to the pixels for changing the bit for each frame is required. This means that, there should be a new design for the pixel, read-out and measurement architecture. We propose a camera structure which is capable of performing both FL and PL implementations using binary random coefficients.

5.4.1 Pixel structure

The conventional APS with integrator structure can be used for the design of the pixel. This structure has been used in chapter 2 to develop the current-mode CS-CMOS imager. Figure 5.4 shows the general schematic of the pixel which is a combination of an APS with integrator design and two extra transistors as the current switches.

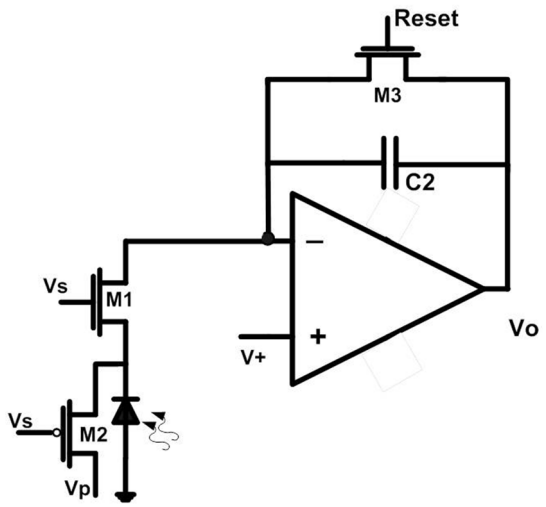


Figure 5.4: General schematic for the designed pixel.

The pixel is capable of integrating the light or keeping the pixel value depending on the value of its *select* signal, i.e., V_s. V_s changes during the integration time corresponding to the pixel's random sequence. After each reset and when V_s is high, the incoming light is integrated in C_2 and when V_s is low, transistor M_2 bypasses the photocurrent from the capacitor, C_2, thus keeping the total charge which is integrated from the beginning to that instant. After the integration of the light for n frames and read-out of the measurement value, the reset signal goes high to discharge the capacitor and a new acquisition for a new

segment of the video starts.

The Op-Amp and capacitor are in-pixel elements. Therefore, the design of the Op-Amp and choosing an appropriate size for the capacitor are of high importance in determining the fill-factor of the design. Also, to put the pixels together and make an imager with any desired size, the read-out strategy and the way of applying *select* signals to the pixels should be appropriately considered in the design of the pixel.

5.4.2 Array architecture

LFSR circuits can be used to provide the select signals for different pixels of the array. However, using one LFSR for the whole array is not area-efficient as the size of the LFSR increases with the number of the pixels while the area devoted to the LFSR can be used to increase the resolution of the camera. Also, the individual access to all pixels for applying the select signals is impractical because of the restrictions in layout design.

The block implementation was used in previous chapters to solve the problem of individual access to the pixels. However, for the video coding in high frame rate applications, it is important to apply the random coefficients to all pixels of the array at the same time. Otherwise, the measurement values for different blocks correspond to different frames of the video.

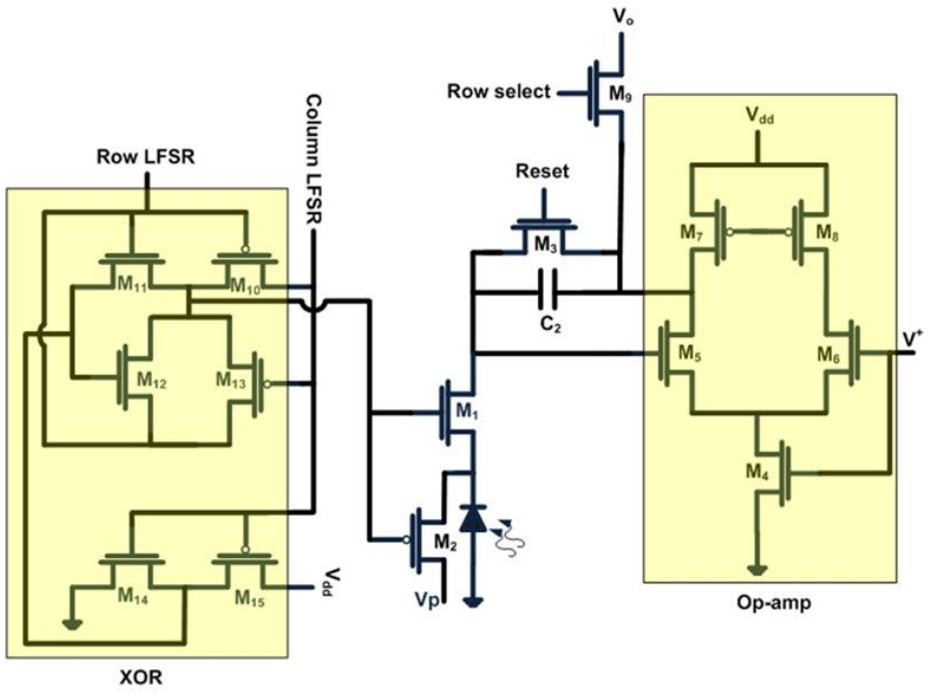
To provide the random bit for each pixel of the array, separate random sequences can be used for the rows and columns of the array. Therefore, there are two random bits for each pixel that correspond to its row and column numbers. Also, an extra XOR is added to the pixel to make a random bit from two bits devoted to each pixel. Two LFSRs for columns and rows of the array provide the required random sequences. Therefore, for the array with $n_1 \times n_2$ pixels, two LFSRs with n_1 and n_2 bits are used instead of one LFSR with $n_1 \times n_2$

bits.

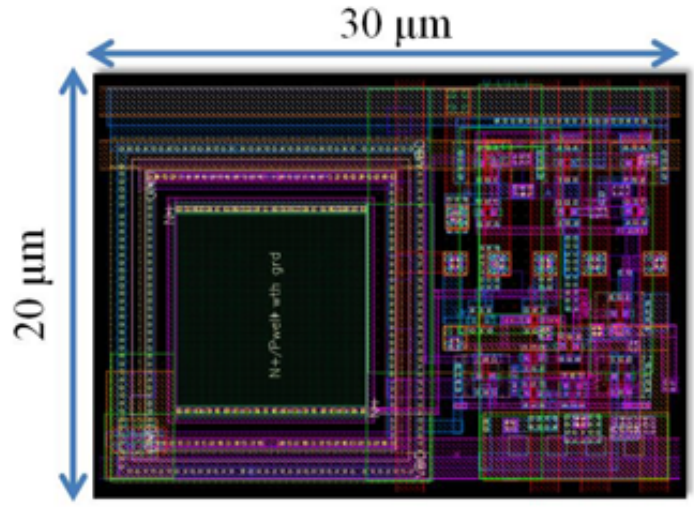
Figure 5.5 shows the schematic and the layout of the designed pixel. The pixel includes an Op-Amp, a capacitor, an XOR gate, four transistors and the photodiode. The size of the pixel is $20\mu m \times 30\mu m$ with photosensitive area of $10\mu m \times 10\mu m$. The fill-factor is $\sim 17\%$. To make the design scalable in terms of read-out structure, the array level (AL) scheme [102] that uses the row select signals and column read-out circuits can be used. The outputs of all pixels in one column are connected together and different rows of pixels are read out one by one when their row select signals are high. Figure 5.6 shows the general schematic of the array with row and column LFSRs and the row and column addressing circuits for the read-out. The LFSRs are working during the light integration while the addressing circuits are functional for the read-out of the measurement values.

5.4.3 Simulation results

Figure 5.7 (a) shows the simulation results for the designed pixel, for different *select* sequences and same photocurrent. The results are for integrating of 10 following frames in integration time of $T_{int} = 250\mu sec$. It leads to $25\mu sec$ for each frame. When the *select* sequence is "0000000000", none of the frames are selected for the measurement and the output is constant regardless of the level of the light. For the *select* sequence of "1010101010", the light is integrated only for 5 frames out of 10. Finally, for "1111111111" sequence, the output is always increasing which means the integration of the light for 10 frames. As



(a)



(b)

Figure 5.5: (a) Schematic diagram of the pixel, (b) The layout screenshot of a single pixel.

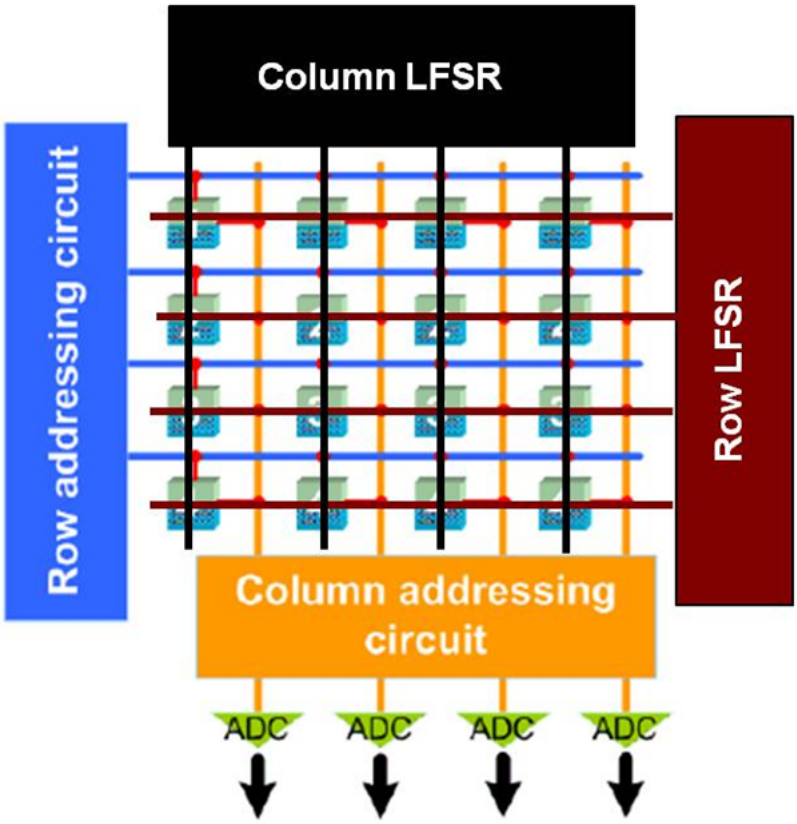
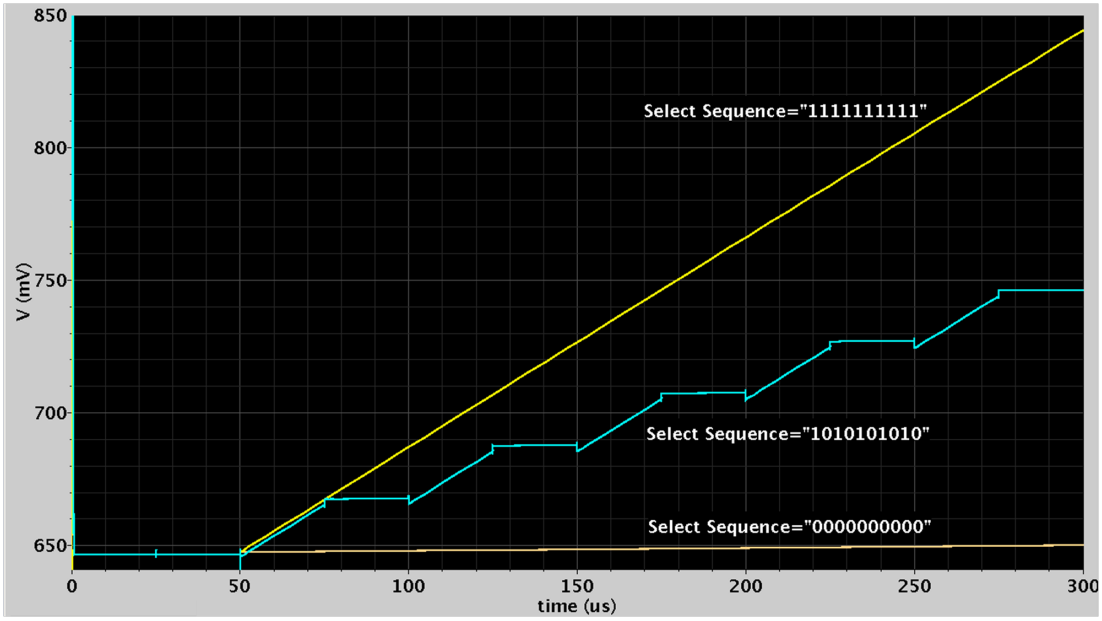


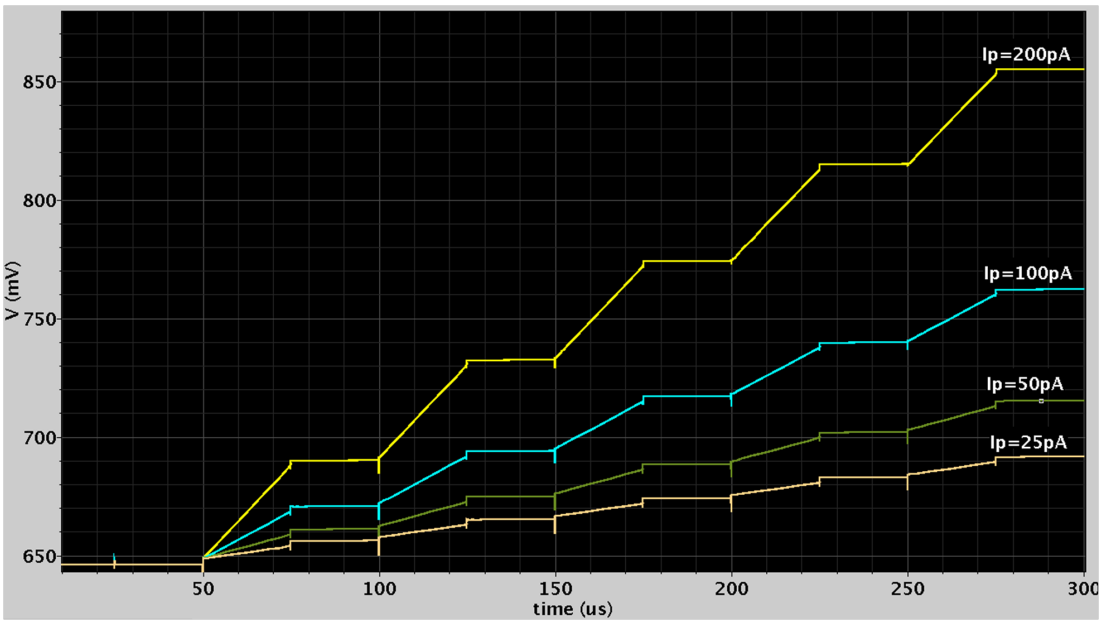
Figure 5.6: Array interconnections for random *select* signals and read-out circuitry.

can be seen in the figure, by increasing the number of selected frames, i.e., number of one bits, the measurement value for the pixel proportionally increases.

Figure 5.7 (b) shows the effect of the different levels of the light, i.e., photocurrent, for the same *select* sequence. As can be seen in the graphs, by increasing the level of the light, the output correspondingly increases.



(a)



(b)

Figure 5.7: Pixel output for (a) 100 pA of photocurrent and different *select* sequences, and (b) same select sequence and different photocurrent values.

5.4.4 Experimental results

The design was fabricated in 130*nm* standard CMOS technology to measure the performance of the pixel. The output of the pixel for different row and column random sequences, leading to different *select* sequences for the pixel, has been measured. The linearity of the measurement values with respect to the level of light and number of selected frames, and also the noise performance of the pixel, were tested.

The screenshot of the oscilloscope for the output of the pixel, for different *select* sequences and $n = 8$ is shown in Figure 5.8. During the integration time, 8 frames can contribute in the measurement values. The lower traces are the output of the XOR gate which is used to select the pixel during the time interval of different frames. It can be seen that when the select signal is high, the light is integrated and the output is increasing. When the select signal in zero, the output is constant and the pixel keeps its value until the next interval for the next frame.

Figure 5.9 shows the measurement values for different levels of light, two types of *select* sequences, and $n = 2, 4, 8$. The integration time, T_{int} is the same for all measurements and the time interval for each frame, i.e., the width of the bit in the sequence, is equal to $\frac{T_{int}}{n}$. As can be seen in the figure, for all different graphs, the measurement values increase linearly with the level of light.

The first type of *select* sequence is when all bits are one. Therefore, the output is always increasing. In this case, for all values of n, the measurement value is the same and there is one curve for all values of n. It is because the integration time is the same for different values of n. However, the larger n leads to the larger number of reconstructed frames captured during the integration time.

The other type of sequence is when only $\frac{n}{2}$ of the bits are one in an n-bit *select* sequence.

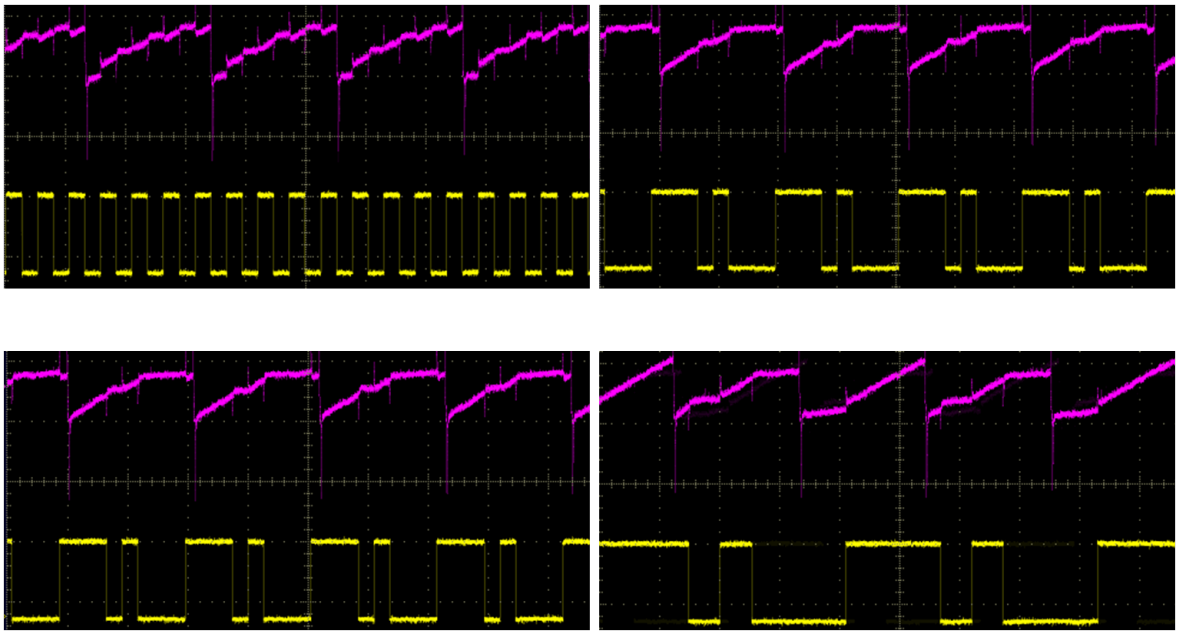


Figure 5.8: Output of the pixel (upper traces) and output of the XOR, i.e., *select* signal, (lower traces) for 4 different *select* signals.

It means that, regardless of the value of n, the light is integrated in half of the integration time. Therefore, the measurement values are half as much as the in the case of all-one sequence. It confirms the monotonicity of the measurement values with the number of light integration interval, i.e., number of ones in the *select* sequence.

Considering the situation when half of the bits are one, then the values are expected to be the same for different n. However, as can be seen in Figure 5.9, there are small differences between the values for different n. This error comes from the switching between '0' and'1' bits during the integration time. The number of these switching events is proportional to the n, thus leading to different switching errors. As can be seen in the figure, this error is less than $5mV$ which is negligible compared to the measurement values.

Figure 5.10 shows the signal-to-noise (SNR) of the measurement values for two different

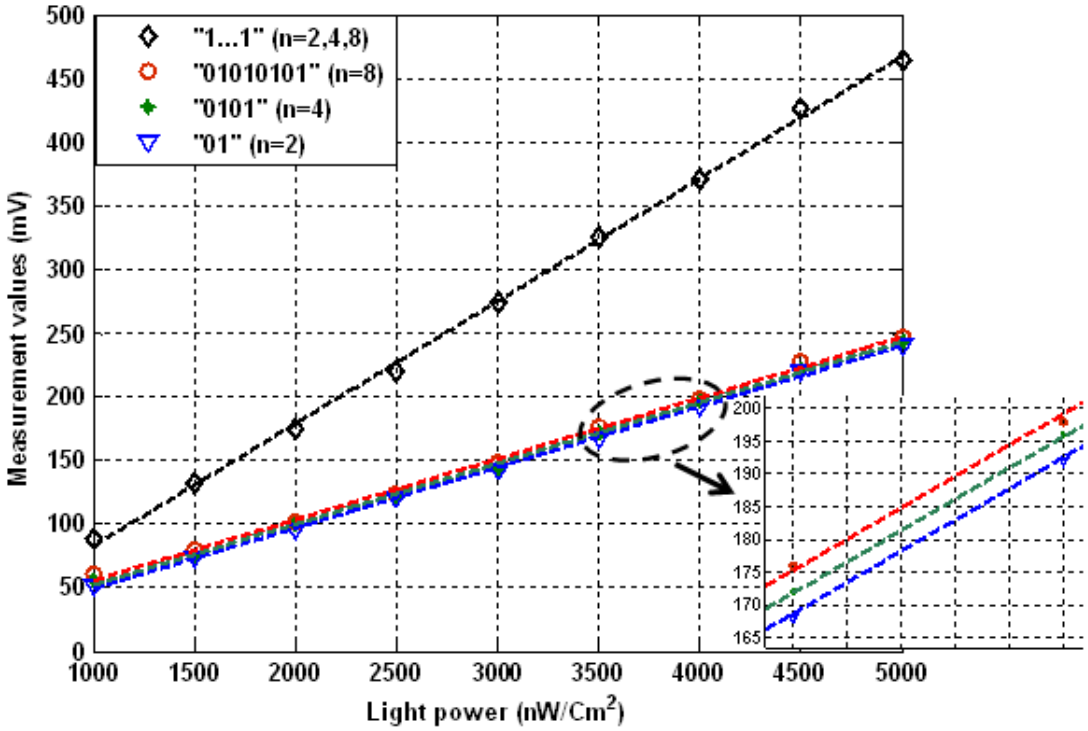


Figure 5.9: Measurement values for different levels of light, for $n = 2, 4, 8$ and two type of *select* sequences. For the first type, all bits and for the second type half of the bits are one.

select sequences. The output of the pixel has been measured for more than 50 different samples for each level of the light. The mean of the samples has been considered as the signal and the variance as the noise. It can be seen that the SNR increase with level of the light, and is higher for the all-one *select* sequence. This is because of the higher level of the signal.

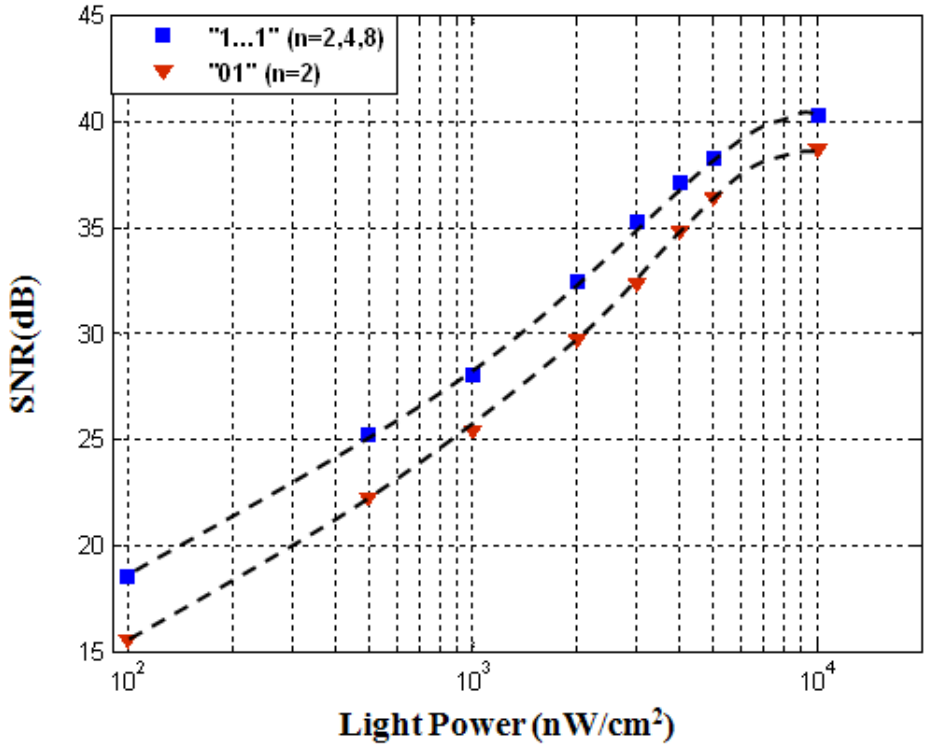


Figure 5.10: Signal-to-noise ratio (SNR) for different levels of the light and different *select* sequences.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The compressive sensing (CS) image/video acquisition for CMOS sensors was considered in this work. Different applications of CMOS sensors which require low-power and high-speed acquisition were introduced. Then, the importance of image compression for imaging systems was discussed. Large image frames translate to a huge amount of data to be stored or sent off the camera. The amount of the data increases with the spatial resolution and frame rate of the camera. Image compression algorithms are used to remove the redundant information of the image data and improve the speed of imaging. Therefore, different approaches in implementation of image compression for CMOS sensors were studied in chapter 1, and compressive acquisition was introduced as an effective way of imaging.

As a compressive acquisition method, CS presents simultaneous compression and sensing of the image by a relatively straightforward encoding process. Different implementations of CS in optical domain by using DMD arrays, coded aperture or random lens systems in early implementations of CS encoding were reviewed. However, the complexity of the optics and the size of the camera make the optical implementation of the CS encoding

unsuitable for portable applications. Also, because of the significant advances in CMOS technology, the implementation of CS on focal plane of the CMOS sensors is very feasible. Therefore, in this thesis, after studying the image reconstruction problem of the CS, we elaborated on real-time and area-efficient implementation of CS method for CMOS image sensors.

In chapter 2, different methods for the CS reconstruction were studied. We elaborated on the basic pursuit as an approach for image reconstruction in 1-norm and total variation minimization methods. Sparse expansions of the image, i.e., DCT, DWT or contourlet transform, were exploited to add new constraints to the reconstruction problem and to remove irrelevant vectors from the feasible set of the optimization problem. Also, one progressive thresholding method was proposed to appropriately find the thresholds used in constraints of the problem. Simulation results showed that modified algorithms are superior to conventional ones, and exploiting contourlet transform results in more improvement because of its ability to detect contours and special geometrical structures in the image. The rest of the thesis was about hardware implementation of the CS encoding process for CMOS sensors. To have an implementation which is compatible with the CS concept, the entire encoding process should be performed during light integration. Therefore, the main objective in designing the CS-CMOS imager is to sense the CS measurement values instead of the pixel values. In this case, extra storage area and time for keeping and manipulating the pixel values are not required, and the entire encoding process is faster. To perform the real-time implementation of the CS, adding extra in-pixel elements might be necessary. However, non-photosensitive, in-pixel elements reduce the fill-factor of the imager which is not desired especially for applications with high sensitivity requirements. In our implementations we avoided the use of in-pixel elements or extra on-array circuitry, while there is no need

to consume time for reading the pixels before making the CS measurements and all CS measurements are directly captured from the array. However, there are some practical restrictions in designing a real-time, and area-efficient CS-CMOS sensor. To overcome these practical issues in designing the layout of the imager, a block-based implementation was performed in our designs.

A current-based, highly-linear implementation of CS was presented in chapter 3. The proposed structure has been implemented in analog domain by using the idea of APS with an integrator and in-pixel current switches. Using a block-based implementation, the integrator circuit was moved out of the array to avoid the use of very complex, low fill-factor pixels. Since, the encoding process is performed during the integration interval, the process is real-time and all CS measurements are directly captured from the array. Also, because of the linearity of the APS with integrator structure, the CS measurement circuits provide a relatively high linearity with respect to the level of the light. The design was fabricated in standard 130nm CMOS technology for two different block sizes. The experimental results for 2×2 and 4×4 blocks confirm the validity and linearity of the measurement process. The functionality of block read-out strategy has been evaluated by fabricating a 16×16 array.

The 3-transistor APS structure is the most commonly used APS structure. It is because the charge-to-voltage conversion is performed inside the pixel while the structure is simple by using only 3 in-pixel transistors. A voltage-based implementation of CS using 3-transistor APS pixels and switched capacitor (SC) circuits as analog adders was presented in chapter 4. Although the charge-to-voltage conversion is performed inside the pixels, the resulting voltage from different pixels are added together during the conversion time, i.e., integration time. This makes the process completely real-time and concurrent with the sensing. Also,

our proposed block read-out approach allows for the scalability of the design and makes it possible to move the extra circuits out of the array. Since there are no extra in-pixel elements compared to the APS structures without compression, then there is no penalty in terms of the fill-factor in our implementation of CS encoding. This design was also fabricated in standard 130nm CMOS technology, and the results were validated for 2×2 and 4×4 block sizes and a 16×16 array.

The current-mode design is more area-efficient as there are two transistors inside each pixel compared with 3 transistors in voltage-mode design. Also, as the sense node voltage of the photodiodes is constant during the integration, all pixels use the same integrating capacitor, and the measurement process is the direct integration of the photocurrent for different pixels, so the encoding process is very linear. However, in the voltage-mode implementation, because of the variable voltage on the sense node, non-uniform switching errors for different levels of the signal, and the process variation for capacitors in different SC branches, the encoding process is less linear compared with the current mode design. On the other hand, in the voltage-mode implementation, the charge-to-voltage conversion is performed inside the pixel. It makes the design more reliable in terms of noise performance. The SNR results in chapter 3 and 4 for both designs show that the voltage-mode design has higher SNR values. Also, the design is functional for lower level of the light compared to the current-mode design where there is no in-pixel conversion or amplification.

The CS-CMOS imagers can also be used for video coding by encoding different frames separately. However, considering temporal correlation between following frames, then the CS measurement can be performed in time domain for a set of image frames instead of individual frames. It decreases the number of read-out intervals, thus leading to a higher frame rate. In chapter 5 a new pixel and array structure design for temporal CS measurement

for video coding using multiple cameras were presented. By adding some more digital circuit to the APS with integrator structure, and using column-row random coefficients, the feasibility of the hardware implementation of multiple-camera CS video coding scheme was confirmed. The pixel was designed and fabricated in 130nm standard CMOS technology, and the measurement results show the functionality of the design in making linear and monotonic measurements.

6.2 Recommendations

The proposed design for CS implementation was implemented for an imager with small array size, and the feasibility of the CS measurement process and the scalability of imager were validated by experimental results. However, the following issues could be considered in development of a complete CS-CMOS imager.

While this work is the general case of the CS-CMOS implementation, the design can be improved and customized for specific applications. Therefore, considering the requirement of each application, and the properties of the target image, then the method of measurement circuitry and reconstruction methods can be modified.

Avoiding the use of complicated in-pixel circuitry was one of the main focus of this work. It makes the design more sensitive, especially for low-level light applications. Also, the low-level light measurements was performed for our design to make sure that the design is functional for low level as well as the high level of illumination. However, there are more flexibility in adding in-pixel components for applications with high levels of illumination. Therefore, the measurement circuits can be placed inside the imager blocks and be shared among different pixels. Also, extra in-pixel assets like in-situ analog memories can be added to the pixel structure. Therefore, there will be more flexibility in changing the block

size, and concurrent measurement extraction for different blocks. This will lead to higher frame rates for the imager.

In the multiple-camera video coding design, the functionality of the pixel for making appropriate measurement were investigated. However, the placement of several cameras in front of the scene such that all cameras make the CS measurement from the exact same scene, is an important issue in development of the multiple-camera video coding system. Therefore, to validate the functionality of the design in making high frame rate videos, the design should be implemented for an array of pixels in order to make the CS measurement from a real scene with high resolution cameras. Also, the the effect of the errors in CS measurements from different cameras because of the incomplete placement of the cameras should be considered and the reconstruction problems should be modified to alleviate this effect.

In our work, we focused on the CS encoding step, while the imaging process in completed after the decoding. The decoding part of CS could be time-consuming, which is unsuitable for real-time imaging systems. Therefore, the hardware implementation of some low-complexity and fast decoding methods for real-time and portable applications is of high importance and should be considered in a future work.

The computation load of the reconstruction can be divided among different processing units to improve the decoding time using parallel processing. This parallel computing can be performed on powerful servers with high computational power. However, it would be helpful to implement the parallel architecture in hardware domain to be used in portable applications.

Bibliography

[1] B. J. Hosticka, W. Brockherde, A. Bumann, T. Heimann, R. Jeremias, A. Kemna, C. Nitta, and O. Schrey, "CMOS Imaging for Automotive Applications," *IEEE Transactions on Electron Devices*, vol. 50, pp. 173–183, 2003.

[2] "http://www.bmva.org/apps/?id=apps."

[3] "http://www.cognex.com/uploadedImages/Company_Information/Fast_Facts/Vision_ Diagram_4col.jpg."

[4] "http://upload.wikimedia.org/wikipedia/commons/3/36/Lane_Departure_Warning.jpg."

[5] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, pp. 1–22, 2009.

[6] E. Culurciello and A. G. Andreou, "CMOS image sensors for sensor networks," *Analog Integrated Circuits and Signal Processing*, vol. 49, pp. 39–51, 2006.

[7] "http://www2.ece.ohio-state.edu/ ekici/images/mwsn.jpg."

[8] G. Feng, P. Swain, and T. Mills, "Wireless endoscopy," *Gastrointestinal Endoscopy*, vol. 51, pp. 725–729, 2000.

[9] M. Pennazio, "Capsule endoscopy: Where are we after 6 years of clinical use?" *Digestive and Liver Disease*, vol. 38, pp. 867–878, 2006.

[10] A. Karargyris and N. Bourbakis, "Wireless capsule endoscopy and endoscopic imaging: A survey on various methodologies presented," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, pp. 72–83, 2010.

[11] T. H. Khan and K. A. Wahid, "Low power and low complexity compressor for video capsule endoscopy," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 1534–1546, 2011.

[12] J. Wang, "Survey and summary from DNA biosensors to gene chips," *Nucleic Acids Research*, vol. 28, pp. 3011–3016, 2000.

[13] L. Benini, C. Guiducci, and C. Paulus, "Electronic detection of DNA hybridization: toward CMOS microarrays," *IEEE Design and Test of Computers*, vol. 24, pp. 38–48, 2007.

[14] S. Parikh, G. Gulak, and P. Chow, "A CMOS image sensor for DNA microarrays," *IEEE Custom Intergrated Circuits Conference (CICC)*, pp. 821–824, 2007.

[15] "www.ist-brighter.eu/tuto14/ANDERSSON/ANDERSSON.pdf."

[16] D. Elson, S. Webb, J. Siegel, K. Suhling, D. Davis, J. Lever, D. Phillips, A. Wallace, and P. French, "Biomedical applications of fluorescence lifetime imaging," *Optics and Photonics News*, vol. 13, pp. 26–32, 2002.

[17] N. Faramarzpour, M. El-Desouki, M. J. Deen, Q. Fang, S. Shirani, and L. W. C. Liu, "CMOS imaging for biomedical applications," *IEEE Potentials*, vol. 27, pp. 31–36, 2008.

[18] "http://www.columbia.edu/~bo8/undergraduate_research/projects/sahil_mehta_-project/images/cDNA-array.jpg."

[19] "http://eng.thesaurus.rusnano.com/upload/iblock/019/small_microarray2.gif."

[20] "Fluorescence imaging principles and methods handbook," *Duke Cancer Center*, Durham, 2002.

[21] K. Sayood, "Introduction to data compression," *Morgan Kaufmann Publishers*, New York, 1998.

[22] N. Ranganathan and S. Henriques, "High-speed VLSI designs for Lempel-Ziv-based data compression," *IEEE Transactions on Circuits and Systems*, vol. 40, pp. 96–106, 1993.

[23] S. G. Miaou and W. S. Chung, "A hardware-oriented gold-washing adaptive vector quantizer and its VLSI architectures for image data compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 1502–1513, 2000.

[24] H. S. Kim, J. Lee, H. Kim, S. Kang, and W. C. Park, "A lossless color image compression architecture using a parallel Golomb-Rice hardware CODEC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, pp. 1581–1587, 2011.

[25] Y. Nishikawa, S. Kawahito, M. Furuta, and T. Tamura, "A high-speed CMOS image sensor with on-chip parallel image compression circuits," *IEEE Conference on Custom Integrated Circuits*, pp. 833–836, 2007.

[26] 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, pp. 538–547, 2011.

[27] S. Kawahito, M. Yoshida, M. Sasaki, K. Umehara, Y. T. D. Miyazaki, 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. 2030–3041, 1997.

[28] 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.

[29] 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.

[30] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "MATIA: A Programmable 80 μW/frame CMOS block matrix transform imager architecture," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 663–672, 2006.

[31] N. Cottini, L. Gasparini, M. D. Nicola, N. Massari, and M. Gottardi, "A CMOS ultra-low power vision sensor with image compression and embedded event-driven energy-management," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, pp. 299–307, 2011.

[32] R. Njuguna and V. Gruev, "Low power programmable current mode computational imaging sensor," *IEEE Sensors Journal*, vol. 12, pp. 727–736, 2012.

[33] J. Fernandez-Berni, R. Carmona-Galan, and L. Carranza-Gonzalez, "FLIP-Q: A QCIF resolution focal-plane array for low-power image processing," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 669–680, 2011.

[34] Z. Lin, M. W. Hoffman, N. S. Walter, D. Leon-Salas, and S. Balkir, "A CMOS image sensor for multi-level focal plane image decomposition," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, pp. 2561–2572, 2008.

[35] M. Zhang and A. Bermak, "Compressive acquisition CMOS image sensor: From the algorithm to hardware implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Sestems*, vol. 18, pp. 490–500, 2010.

[36] M. Zhang and A. Bermak, "Quadrant-based online spatial and temporal compressive acquisition for CMOS image sensor," *IEEE Transactions on Very Large Scale Integration (VLSI) Sestems*, vol. 19, pp. 1525–1534, 2011.

[37] M. A. Herman and T. Strohmer, "High-resolution radar via compressed sensing," *IEEE Transactions on Signal Processing*, vol. 57, pp. 2275–2284, 2009.

[38] A. C. Gurbuz, J. H. McClellan, and W. R. Scott, "A compressive sensing data acquisition and imaging method for stepped frequency GPRs," *IEEE Transactions on Signal Processing*, vol. 57, pp. 2640–2650, 2009.

[39] M. A. Sheikh, O. Milenkovic, S. Sarvotham, and R. G. Baraniuk, "Compressed sensing DNA microarrays," *ECE-Rice University, TREE0706*, 2007.

[40] F. Parvaresh, H. Vikalo, S. Misra, and B. Hassibi, "Recovering sparse signals using sparse measurement matrices in compressed DNA microarrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, pp. 275–285, 2008.

[41] J. Ma, "Compressed sensing for surface characterization and metrology," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, pp. 1600–1615, 2010.

[42] G. Wang, Y. Bresler, and V. Ntziachristos, "Guest editorial compressive sensing for biomedical imaging," *IEEE Transactions on Medical Imaging*, vol. 30, pp. 1013–1016, 2011.

[43] M. Lustig, D. Donoho, and J. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance in Medicine*, vol. 58, pp. 1182–1195, 2007.

[44] M. Lusting, D. Donoho, J. Santos, and J. Pauly, "Compressed sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, pp. 72–82, 2008.

[45] T. Cukur, M. Lusting, E. Saritas, and D. Nishimura, "signal compensation and compressed sensing for magnetization-prepared MR angiograph," *IEEE Transaction on Medical Imaging*, vol. 30, pp. 1017–1027, 2011.

[46] G. H. Chen, J. Tang, and S. Leng, "Prior image constrained compressed sensing (PICCS): A method to accurately reconstruct dynamic CT images from highly undersampled projection data sets," *Medical Physics*, vol. 35, pp. 660–663, 2008.

[47] E. Y. Sidky and X. Pan, "Image reconstruction in circular conebeam computed tomography by constrained, total-variation minimization," *Physics in Medicine and Biology*, vol. 53, pp. 4777–4807, 2008.

[48] H. Yu and G. Wang, "Compressed sensing based interior tomography," *Physics in Medicine and Biology*, vol. 54, pp. 2791–2805, 2009.

[49] C. Quinsac, A. Basarab, J. Girault, , and D. Kouame, "Compressed sensing of ultrasound images: Sampling of spatial frequency domains," *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 231–236, 2010.

[50] A. Achim, B. Buxton, G. Tzagkarakis, and P. Tsakalides, "Compressive sensing for ultrasound and RF echoes using alpha-stable distributions," *32nd Annual IEEE International Conference on Engineering in Medicine and Biology*, pp. 4304–4307, 2010.

[51] I. Tosic, I. Jovanovic, P. Frossard, M. Vetterli, and N. Duric, "Ultrasound tomography with learned dictionaries," *IEEE International Conference on Acoustics, Speech Signal Processing (ICASSP)*, pp. 5502–5505, 2010.

[52] A. F. Coskun, I. Sencan, T. Su, and A. Ozcan, "Lensless wide-field fluorescent imaging on a chip using compressive decoding of sparse objects," *Optics Express*, vol. 18, pp. 10 510–10 523, 2010.

[53] M. F. Durate, M. A. Devenpor, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 83–91, 2008.

[54] "Rice Single-Pixel Camera Project, http://dsp.rice.edu/cscamera."

[55] J. Samspell, "Digital micromirror device (DMD) and its application to projection displays," *Journal of Vacuum Science and Technology B*, vol. 12, pp. 3242–3246, 1994.

[56] P. Ye, J. L. Paredes, Y. Wu, C. Chen, G. R. Arce, and D. W. Parther, "Compressive

confocal microscopy: 3D reconstruction algorithms," *Proceeding of SPIE*, vol. 7210, pp. 72 100G1–72 100G12, 2009.

[57] Y. Wu, P. Ye, I. O. Mirza, G. R. Arce, and D. W. Parther, "Experimental demonstration of an optical-sectioning compressive sensing microscope (CSM)," *Optics Express*, vol. 18, pp. 24 565–24 578, 2010.

[58] W. L. Chan, K. Charan, D. Takhar, K. F. Kelly, R. G. Baraniuk, and D. M. Mittleman, "A single-pixel terahertz imaging system based on compressed sensing," *Applied Physics Letter*, vol. 93, pp. 121 105–121 105–3, 2008.

[59] P. K. Baheti and M. A. Neifeld, "Feature-specific structured imaging," *Applied Optics*, vol. 45, pp. 7382–7391, 2008.

[60] S. R. Gottesman and E. E. Fenimore, "New family of binary arrays for coded aperture imaging," *Applied Optics*, vol. 28, pp. 4344–4352, 1989.

[61] R. F. Marcia and R. M. Willet, "Compressive coded aperture superresolution image reconstruction," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 833–836, 2008.

[62] A. Wagadarikar, R. John, R. Willet, and D. Brady, "Single disperser design for coded aperture snapshot spectral imaging," *Applied Optics*, vol. 47, pp. B44–B51, 2008.

[63] H. Arguello, H. F. Rueda, and G. R. Arce, "Spatial super-resolution in code aperture spectral imaging," *Proceeding of SPIE*, vol. 8365, pp. 83 650A–1, 2012.

[64] A. Stern and B. Javidi, "Random projections imaging with extended space-bandwidth product," *Journal of Display Technology*, vol. 3, pp. 316–320, 2007.

[65] R. Fergus, A. Torralba, and W. T. Freeman, "Random lens imaging," *MIT-CASIL Technical Report*, 2006.

[66] M. Zhang, Y. Wang, and A. Bermak, "Block based compressive sampling for digital pixel sensor array," *2nd Asia Symposium on Quality Electronic Design (ASQED)*, pp. 9–12, 2010.

[67] Y. Oike and A. E. Gamal, "A 256×256 CMOS image sensor with $\Delta\Sigma$-based single-shot compressed sensing," *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 386–388, 2012.

[68] V. Majidzadeh, L. Jacques, A. Schmid, P. Vandergheynst, and Y. Leblebic, "A (256×256) pixel 76.7mW CMOS imager/compressor based on real-time in-pixel compressive sensing," *Proceeding of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2956–2959, 2010.

[69] R. Robucci, J. D. Gray, L. K. Chiu, J. Romberg, and P. Hasler, "Compressive sensing on a CMOS separable-transform image sensor," *Proceeding of the IEEE*, vol. 98, pp. 1089–1101, 2010.

[70] L. Xiao, K. Liu, and D. Han, "CMOS low data rate imaging method based on compressed sensing," *Optics & Laser Technology*, vol. 44, pp. 1338–1345, 2012.

[71] Y. M. Chi, A. Abbas, S. Chakrabartty, and G. Cauwenberghs, "An active pixel CMOS separable transform image sensor," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1281–1284, 2009.

[72] L. Jacques, P. Vandergheynst, A. Bibet, V. Majidzadeh, A. Schmid, and Y. Leblebici,

"CMOS compressed imaging by random convolution," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1113–1116, 2009.

[73] M. R. Dadkhah, M. J. Deen, and S. Shirani, "Block-based compressive sensing in a CMOS image sensor," *IEEE Sensors Journal*, 2013.

[74] M. R. Dadkhah, M. J. Deen, and S. Shirani, "CMOS image sensor with area-efficient block-based compressive sensing," unpublished, 2013.

[75] A. E. Gamal and H. Eltoukhy, "CMOS image sensors," *IEEE Circuits and Devices Magazine*, vol. 21, pp. 6–20, 2005.

[76] M. R. Dadkhah, M. J. Deen, and S. Shirani, "Compressive Sensing Image Sensors-Hardware Implementation," *Sensors*, vol. 13, pp. 4961–4978, 2013.

[77] M. R. Dadkhah, S. Shirani, and M. J. Deen, "Compressive sensing with modified total variation minimization algorithm," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1310–1313, 2010.

[78] M. R. Dadkhah, M. J. Deen, and S. Shirani, "CMOS Sensors for Compressive Sensing," *The Electrochemical Society, Meeting Abstracts*, p. 805, 2012.

[79] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, pp. 118–124, 2007.

[80] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, pp. 4203–4215, 2007.

[81] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus mathematique*, vol. 349, pp. 589–592, 2008.

[82] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 21–30, 2008.

[83] S. Boyd and L. Vandenberghe, "Convex Optimization," *Cambridge University Press*, New York, 2004.

[84] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, 2006.

[85] J. Romberg, "Imaging via compressive sampling [introduction to compressive sampling and recovery via convex programming]," *IEEE Signal Processing Magazine*, vol. 25, p. 1420, 2008.

[86] Y. Nestrov and A. Nemrovski, "Interior point polynomial time methods in convex programming," *Studies in Applied and Numerical Mathematics (SIAM)*, Philadelphia, 1994.

[87] M. A. T. Figueiredo, R. D. Nowark, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, pp. 586–597, 2007.

[88] J. M. Bioucas-Dias and M. A. T. Figueiredo, "A New TwIST: Two step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image Processing*, vol. 16, pp. 2992–3004, 2007.

[89] S. J. Wright, R. D. Nowark, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, pp. 2479–2493, 2009.

[90] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, pp. 4655–4666, 2007.

[91] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 58, pp. 1094–1121, 2012.

[92] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 310–316, 2010.

[93] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, pp. 2230–2249, 2009.

[94] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate sapmles," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, 2009.

[95] H. Wu and S. Wang, "Adaptive sparsity matching pursuit algorithm for sparse reconstruction," *IEEE Signal Processing Letters*, vol. 19, pp. 471–474, 2009.

[96] A. Borghi, J. Darbon, S. Peyronnet, T. F. Chan, and S. Osher, "A simple compressive sensing algorithm for parallel many-core architectures," *Journal of Signal Processing Systems*, vol. 71, pp. 1–20, 2013.

[97] D. Schneider, "New camera chip captures only what it needs," *IEEE Spectrum*, vol. 50, pp. 13–14, 2013.

[98] M. N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Transactions on Image Processing*, vol. 14, pp. 2091–2106, 2005.

[99] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 45, pp. 458–560, 1992.

[100] N. Faramarzpour, M. J. Deen, S. Shirani, Q. Fang, L. W. C. Liu, and F. de Souza Campos, "CMOS based active pixel for low-light-level detection: analysis and measurements," *IEEE Transactions on Electron Devices*, vol. 54, pp. 3229–3237, 2007.

[101] N. Faramarzpour, M. El-Desouki, M. J. Deen, S. Shirani, and Q. Fang, "CMOS photodetector systems for low-level light applications," *J. Mater. Sci.: Mater. Electron.*, vol. 20, pp. S87–S93, 2009.

[102] M. El-Desouki, M. J. Deen, Q. Fang, L. Liu, F. Tse, and D. Armstrong, "CMOS image sensors for high speed applications," *Sensors*, vol. 9, pp. 430–444, 2009.

[103] "http://www.photron.com/index.php?cmd=gallery."

[104] S. T. Thorddsen, T. G. Etoh, and K. Takehara, "High-speed imaging of drops and bubbles," *Annu. Rev. Fluid Mech.*, vol. 40, pp. 257–285, 2008.

[105] G. M. Gibson, J. Leach, S. Keen, A. J. Wright, and M. J. Padgett, "Measuring the accuracy of particle position and force in optical tweezers using high-speed video microscopy," *Optics Express*, vol. 16, pp. 14 561–14 570, 2008.

[106] N. Seto, S. Katayama, and A. Matsunawa, "High-speed simultaneous observation of plasma and keyhole behavior during high power CO_2 laser welding: Effect of shielding gas on porosity formation," *Journal of Laser Applications*, vol. 12, pp. 245–250, 2000.

[107] G. Settles, "High-speed imaging of shock waves, explosions and gunshots: New digital video technology, combined with some classic imaging techniques, reveals shock waves as never before," *American Scientist*, pp. 22–31, 2006.

[108] M. Vollmer and K. P. Mllmann, " High speed and slow motion: the technology of modern high speed cameras," *Physics Education*, vol. 46, pp. 191–202, 2011.

[109] R. F. Marcia and R. M. Willet, "Compressive coded aperture video reconstruction," *16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland*, 5 pages, 2008.

[110] N. Jacobs, S. Schuh, and R. Pless, "Compressive sensing and differential image-motion estimation," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 718–721, 2010.

[111] X. Wu and R. Pournaghi, "High frame rate video capture by multiple cameras with coded exposure," *IEEE International Conference on Image Processing (ICIP)*, pp. 577–580, 2010.

Appendix A

MATLAB codes for CS reconstruction:

```
To perform the CS reconstruction for different methods
mentioned in chapter 2:
function [CTVIr,CTVPSNR]=mcCS_DCT(I,n,m,e1,e2,e3)
%n is the dimension of DCT (n by n DCT)
%m is the number of measurments
%e1: Noise constraint
%e2 and e3: Thresholds for additional constraints
M=size(I,1);N=size(I,2);
Ir2=zeros(M,N);mIr2=zeros(M,N);cmIr2=zeros(M,N);
load b0; load b1; load b2; load b3;
HH=HHDWT(n,n); phi=vec_2DCT(n);
A1=zeros(n^2);% A1: finding the bottom half of the image
for i=1:n^2
    d=floor(2*(i-.1)/n);
    if mod(d,2)==1
```

```
        A1(i,i)=1;
    end
end
A2=zeros(n^2);% A2: finding the top half of the image
for i=1:n^2
    d=floor(2*(i-.1)/n);
    if mod(d,2)==0
        A2(i,i)=1;
    end
end
A3=[zeros((n^2)/2,n^2);[zeros(n^2/2),eye(n^2/2)]];
% A3: finding the right half of the image
A4=zeros(n^2);
% A4: finding the right-bottom quarter of the image
for i=1:n/2
    A4(1+n^2/2+n/2*(2*i-1):n^2/2+n/2*(2*i),
    1+n^2/2+n/2*(2*i-1):n^2/2+n/2*(2*i))=eye(n/2);
end
TV=TVmatrix(n);
mvec=zeros(1,n^2);
mvec(1)=1;
k=0;
for i=1:M/n
    for j=1:N/n
        k=k+1;
        Ib= I(n*(i-1)+1:n*i,n*(j-1)+1:n*j);
```

```
        load Measurements;
        load Randommatrix;
        y=Measurements(k,:)';
        R=Randommatrix(:,:,k);
%************************Using CVX (Norm1)************************
        % cvx_begin
        % variable xb2(n*n)
        % minimize( norm( phi * xb2 , 1 ) )
        %  subject to
        %   norm(y-R*xb2,2)<e1;%
        %   norm(A4*phi*xb2,2)/(n^2/4)<e2;
        %   norm(TV*xb2,2)/(n^2)<e2;
        %   norm(HH*xb2,2)/(n^2/4)<e2;
        %   norm(b0*xb2,2)/(n^2/4)<e3;
        %   norm(b1*xb2,2)/(n^2/4)<e3;
        %   norm(b2*xb2,2)/(n^2/4)<e3;
        %   norm(b3*xb2,2)/(n^2/4)<e3;
        % cvx_end
        % WN1Ir(n*(i-1)+1:n*i,n*(j-1)+1:n*j)=reshape(xb2,n,n);
        % WN1PSNR=10*log10(max(I(:))^2/(norm(I(:)-WN1Ir(:))^2/M/N));
 %************************Using CVX (TV)************************
        cvx_begin
        variable xb2(n*n)
        minimize( norm( TV * xb2 , 1 ) )
        subject to
        norm(y-R*xb2,2)<e1;
```

```
        %  norm(TV*xb2,2)/(n^2)<e2;
        %  norm(A4*phi*xb2,2)/(n^2/4)<e2;
        %  norm(HH*xb2,2)/(n^2/4)<e2;
        norm(b0*xb2,2)/(n^2/4)<e3;norm(b1*xb2,2)/(n^2/4)<e3;
        norm(b2*xb2,2)/(n^2/4)<e3;norm(b3*xb2,2)/(n^2/4)<e3;
        cvx_end
        CTVIr(n*(i-1)+1:n*i,n*(j-1)+1:n*j)=reshape(xb2,n,n);
    end
end
CTVPSNR=10*log10(max(I(:))^2/(norm(I(:)-CTVIr(:))^2/M/N));
%-----------------------------------------------------------------
To provide the coefficient matrices for contourlet transform:
function [c,b0,b1,b2,b3]=Contourlet(Q0,Q1,hlp,glp,hdfb,M,N,m,n)
[d,c]=LPvec(hlp,glp,M,N,m,n);
[Sv0,Sv1,Sv2,Sv3]=DFB4vecf(M,N,Q0,Q1,'cd');
b0=Sv0*d; b1=Sv1*d; b2=Sv2*d; b3=Sv3*d;
%-----------------------------------------------------------------
To provide the coefficient matrix for wavelet transform:
function HH=HHDWT(M,N)
[h0,h1]=dfilters('cd','d');Iv2=myresample2(2*eye(2),M,N);
[h0,h1]=dfilters('cd','d');h1v=vecfilt(h1,M,N);
G1v=MIvecfilt(h1,M,N,eye(2));HH=Iv2*G1v*h1v;
%-----------------------------------------------------------------
To provide the coefficient matrix for DCT transform:
function A=vec_2DCT(n)
H=zeros(n);Ac=zeros(n*n);
```

```
for m=1:n
    i=m-1;
    for mm=1:n
        j=mm-1;
        if i==0
            H(m,mm)=sqrt(1/n)*cos(i*pi*(2*j+1)/2/n);
        else
            H(m,mm)=sqrt(2/n)*cos(i*pi*(2*j+1)/2/n);
        end
    end
end
H1=[H,zeros(n,n*(n-1))];
for i=1:n
    Ac(n*(i-1)+1:n*i,:)=circshift(H1,[0,(i-1)*n]);
end
He=zeros(n,n^2);
for i=1:n
    He(:,(i-1)*n+1)=H(:,i);
end
Ar=zeros(n*n);
for i=1:n
    for j=1:n
        Ar((i-1)*n+j,:)=circshift(He(i,:),[0,j-1]);
    end
end
A=Ar*Ac;
```

```
%-----------------------------------------------------------------
To provide the coefficient matrix for total variation:
function TV=TVmatrix(n)
TV=zeros(4*n^2,n);TV(3,1)=1;TV(3,2)=-1;TV(4,1)=1;TV(4,1+n)=-1;
for i=2:n^2
    if(floor(i/n)==0 && mod(i,n)~=0)
        TV(4*(i-1)+1,i)=1; TV(4*(i-1)+1,i-1)=-1;
        TV(4*(i-1)+2,i)=1; TV(4*(i-1)+2,i+1)=-1;
        TV(4*(i-1)+3,i)=1;TV(4*(i-1)+3,i+n)=-1;
    end
    if (floor(i/n)>=1)
        if (i==n)
            TV(4*(i-1)+1,i)=1;TV(4*(i-1)+1,i-1)=-1;
            TV(4*(i-1)+2,i)=1;TV(4*(i-1)+2,i+n)=-1;
        else if(floor(i/n)<n-1)
                TV(4*(i-1)+3,i)=1;TV(4*(i-1)+3,i+n)=-1;
                TV(4*(i-1)+4,i)=1;TV(4*(i-1)+4,i-n)=-1;
                if (mod(i,n)~=1 && mod(i,n)~=0)
                    TV(4*(i-1)+1,i)=1;TV(4*(i-1)+1,i-1)=-1;
                    TV(4*(i-1)+2,i)=1;TV(4*(i-1)+2,i+1)=-1;
                else if (mod(i,n)==0)
                        TV(4*(i-1)+1,i)=1; TV(4*(i-1)+1,i-1)=-1;
                    else if(mod(i,n)==1)
                            TV(4*(i-1)+2,i)=1;TV(4*(i-1)+2,i+1)=-1;
                        end
                    end
```

```
                end
            else if ( floor(i/n)>=n-1)
                    TV(4*(i-1)+4,i)=1;TV(4*(i-1)+4,i-n)=-1;
                    if (mod(i,n)~=1 && mod(i,n)~=0)
                        TV(4*(i-1)+1,i)=1;TV(4*(i-1)+1,i-1)=-1;
                        TV(4*(i-1)+2,i)=1;TV(4*(i-1)+2,i+1)=-1;
                    else if (mod(i,n)==0)
                            TV(4*(i-1)+1,i)=1;TV(4*(i-1)+1,i-1)=-1;
                        else if (mod(i,n)==1)
                                TV(4*(i-1)+2,i)=1;
                                TV(4*(i-1)+2,i+1)=-1;
                            end
                        end
                    end
                end
            end
        end
    end
end
%-------------------------------------------------------------------
To provide the coefficients for laplacian pyramid in contourlet transform:
function [d,c]=LPvec(h,g,M,N,m,n)
H=vecfilt2s(h,M,N);G=vecfilt2s(g,M,N);
D=dsample(M,N,m,n);U=usample(M/m,N/n,m,n);
c=D*H;d=eye(M*N)-G*U*c;
%-------------------------------------------------------------------
```

```
To provide the coefficients for digital filters in contourlet transform:
function [Sv0,Sv1,Sv2,Sv3]=DFB4vecf(M,N,Q0,Q1,fname)
Iv2=myresample2(Q1*Q0,M,N);
h=dfilters(fname,'d');hc=modulate2(h, 'c');hr=modulate2(h, 'r');
hcv=vecfilt(hc,M,N);hrv=vecfilt(hr,M,N);
Gcv=MIvecfilt(hc,M,N,Q0);Grv=MIvecfilt(hr,M,N,Q0);
Sv0=Iv2*Grv*hrv;Sv1=Iv2*Gcv*hrv;Sv2=Iv2*Grv*hcv;Sv3=Iv2*Gcv*hcv;
%-------------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
function H=vecfilt2s(h,M,N)
n=numel(h); h1=zeros(n); h1(ceil(n/2),:)=h;H1=vecfilt(h1,M,N);
h2=zeros(n); h2(:,ceil(n/2))=h;H2=vecfilt(h2,M,N); H=H2*H1;
%-------------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
function G=vecfilt(h,M,N)
[n1,n2]=size(h);
if mod(n1,2)==0
    h=[h;zeros(1,n2)];[n1,n2]=size(h);
end
if mod(n2,2)==0
    h=[h,zeros(n1,1)];[n1,n2]=size(h);
end
n11=floor(n1/2);n22=floor(n2/2);o=[ceil(n1/2),ceil(n2/2)];
G=zeros(M*N);
for i=1:M
    for j=1:N
```

```
        temp2=zeros(1,M*N);
        for t1=-n11:n11
            for t2=-n22:n22
                It=i+t1;Jt=j+t2;
                if ((It<1) | (It>M))
                    It=i-t1;
                end
                if ((Jt<1) | (Jt>N))
                    Jt=j-t2;
                end
                k=(Jt-1)*M+It;temp2(k)=temp2(k)+h(o(1)+t1,o(2)+t2);
            end
        end
        k1=(j-1)*M+i;G(k1,:)=temp2;
    end
end
%-----------------------------------------------------------------
Downsample function:
function D=dsample(M,N,m,n)
    D1=zeros(M*N/m,M*N);
    for i=1:M*N/m
        D1(i,(i-1)*m+1)=1;
    end
    D2=zeros(M*N/m/n,M*N/m);T=[eye(M/m),zeros(M/m,M*N/m-M/m)];
    for i=1:n:N
        D2(floor((i-.001)/n)*M/m+1:(floor((i-.001)/n)+1)*M/m,:)=
```

```
        circshift(T,[0,(i-1)*M/m]);
    end
    D=D2*D1;
%-------------------------------------------------------------------
Upsample function:
function U=usample(M,N,m,n)
U1=zeros(M*N*m,M*N);
if m==1
    U1=eye(M*N);
else
    for i=1:M*N*m
        if mod(i,m)==1
            U1(i,floor(i/m)+1)=1;
        end
    end
end
U2=zeros(M*N*m*n,M*N*m);T=[eye(M*m),zeros(M*m,M*N*m-M*m)];j=0;
for i=1:N
    U2((i-1)*n*M*m+1:((i-1)*n+1)*M*m,:)=circshift(T,[0,j*M*m]);
    j=j+1;
end
U=U2*U1;
%-------------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
function Qv=myresample(Q,M,N)
c=1;Qv=zeros(M*N,M*N);
```

```
for j=1:N
    for i=1:M
        k=(j-1)*M+i;t1=(Q)*[M/2-i;j-N/2];
        ii=t1(1);jj=t1(2);Nv=(jj+N/2-1)*M+M/2-ii;
        if ((jj<=N/2)&& (jj>=(1-N/2)) && (ii<=(M/2-1))
            && (ii>=(-M/2))&& (Nv>0) )
            Qv(c,Nv)=1;
        end
          c=c+1;
    end
end
%-----------------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
% Copyright (c) 2009, Minh Do All rights reserved.
function [h0, h1] = dfilters(fname, type)
switch fname
    case {'haar'}
        if lower(type(1)) == 'd'
            h0 = [1, 1] / sqrt(2);h1 = [-1, 1] / sqrt(2);
        else
            h0 = [1, 1] / sqrt(2);h1 = [1, -1] / sqrt(2);
        end
    case {'5-3', '5/3'}
         [h0, g0] = pfilters('5-3');
         if lower(type(1)) == 'd'
             h1 = modulate2(g0, 'c');
```

```
        else
            h1 = modulate2(h0, 'c');h0 = g0;
        end
        t = [0, 1, 0; 1, 0, 1; 0, 1, 0] / 4;
        h0 = mctrans(h0, t);h1 = mctrans(h1, t);
    case {'cd', '9-7', '9/7'}
        [h0, g0] = pfilters('9-7');
        if lower(type(1)) == 'd'
            h1 = modulate2(g0, 'c');
        else
            h1 = modulate2(h0, 'c');h0 = g0;
        end
        t = [0, 1, 0; 1, 0, 1; 0, 1, 0] / 4;
        h0 = mctrans(h0, t);h1 = mctrans(h1, t);
    case {'pkva6', 'pkva8', 'pkva12', 'pkva'}
        beta = ldfilter(fname);[h0, h1] = ld2quin(beta);
        h0 = sqrt(2) * h0;h1 = sqrt(2) * h1;
        if lower(type(1)) == 'r'
            f0 = modulate2(h1, 'b');f1 = modulate2(h0, 'b');
            h0 = f0;h1 = f1;
        end
    otherwise
        error('Unrecognized directional filter name');
end
%-----------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
```

```
% Copyright (c) 2009, Minh Do All rights reserved.
function y = modulate2(x, type, center)
if ~exist('center', 'var')
    center = [0, 0];
end
s = size(x);o = floor(s / 2) + 1 + center;
n1 = [1:s(1)] - o(1);n2 = [1:s(2)] - o(2);
switch lower(type(1))
    case 'r'
     m1 = (-1) .^ n1;y = x .* repmat(m1', [1, s(2)]);
    case 'c'
     m2 = (-1) .^ n2;y = x .* repmat(m2, [s(1), 1]);
    case 'b'
     m1 = (-1) .^ n1;m2 = (-1) .^ n2;
     m = m1' * m2;y = x .* m;
    otherwise
     error('Invalid input type');
end
%---------------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
function G=vecfilt(h,M,N)
[n1,n2]=size(h);
if mod(n1,2)==0
    h=[h;zeros(1,n2)];[n1,n2]=size(h);
end
if mod(n2,2)==0
```

```
    h=[h,zeros(n1,1)];[n1,n2]=size(h);
end
n11=floor(n1/2);n22=floor(n2/2);
o=[ceil(n1/2),ceil(n2/2)];G=zeros(M*N);
for i=1:M
    for j=1:N
        temp2=zeros(1,M*N);
        for t1=-n11:n11
            for t2=-n22:n22
                It=i+t1;Jt=j+t2;
                if ((It<1) | (It>M))
                    It=i-t1;
                end
                if ((Jt<1) | (Jt>N))
                    Jt=j-t2;
                end
                k=(Jt-1)*M+It;
                temp2(k)=temp2(k)+h(o(1)+t1,o(2)+t2);
            end
        end
        k1=(j-1)*M+i;G(k1,:)=temp2;
    end
end
%-------------------------------------------------------------
Auxiliary function to use in contourlet transform implementation:
function G=MIvecfilt(h,M,N,Q)
```

```
sh1=size(h,1);sh2=size(h,2);
hn=zeros(sh1,sh2);
for i=1:sh1
    for j=1:sh2
        temp=inv(Q)*[ceil(sh1/2)-i;j-ceil(sh2/2)];
        ii=-temp(1)+ceil(sh1/2);jj=temp(2)+ceil(sh2/2);
        if (ceil(ii)==ii && ceil(jj)==jj && ii>0 && jj>0
            && ii<=sh1 && jj<=sh2 )
            hn(i,j)=h(ii,jj);
        end
    end
end
G=vecfilt(hn,M,N);
%-----------------------------------------------------------------
```

Appendix B

VHDL codes for the experimental set-up (image coding measurements):

```
library altera;
use altera.altera_primitives_components.all;
LIBRARY ieee;
USE ieee.std_logic_1164.all;
Use Ieee.std_logic_unsigned.all;

-- To provide:
-- 1. Clocking signals required for the LFSR:
--    clk,clkn,clear, and LD.
-- 2. Clocking signals required for the LFSR for block-switch:
--    clk1,clkn1,clear1, and LD1.
-- 3. Reset signal for pixels:
--    RS.
-- 4. Switching clocks fpr switched capacitor circuit:
--    Phi1, and Phi2.
```

```
-- 5. Clocking signals for analog-to-digital converter:
--    A2Dclk, andA2DCS.
-- 6. Signal for serial transmission towards PC:
--    TxD.

ENTITY finaltestbench IS
   PORT ( clkin_board, reset: IN STD_LOGIC;
          A2Dout:IN BIT;
          clk,clkn, Phi1, Phi2, RS, clear,LD,A2Dclk,A2DCS,clk1,
          clkn1,clear1,LD1: OUT STD_LOGIC;
          TXD: out BIT);
END finaltestbench;

ARCHITECTURE Structure OF finaltestbench IS

COMPONENT JR1_clocks
            PORT ( clkin_board,reset: IN STD_LOGIC;
           A2Dout: IN BIT;
           TXD: OUT BIT;
          clk,clkn, Phi1, Phi2, RS, clear,LD,A2Dclk,A2DCS,clk1,
          clkn1,clear1,LD1: OUT STD_LOGIC--loadtemp,conversiontemp
          );
END COMPONENT;
COMPONENT Clkdivider
   PORT ( clkin, resetn: IN STD_LOGIC;
          clkout: OUT STD_LOGIC_VECTOR (7 downto 0));
```

```
   END COMPONENT;
SIGNAL clkt,clknt, Phi1t, Phi2t,RSt, cleart,LDt,A2Dclkt,A2DCSt,
temp,clk1t,clkn1t,clear1t,LD1t: STD_LOGIC;
signal TXDtemp:BIT;
signal a2dout1:BIT;
Signal clktemp1,clktemp2,clktemp3: STD_LOGIC_VECTOR (7 downto 0);
Begin
 test21: JR1_clocks PORT MAP(clkin_board,reset,A2Dout,TXDtemp,clkt
 ,clknt, Phi1t, Phi2t, RSt, cleart,LDt,A2Dclkt,A2DCSt,clk1t,
 clkn1t,clear1t,LD1t);
 clk <= clkt; clkn <= clknt; Phi1 <= Phi1t; Phi2 <= Phi2t;
  RS <= RSt; clear <= cleart; LD <= LDt;  TXD <= TXDtemp;
 A2Dclk <=A2Dclkt; A2DCS <=A2DCSt; LD1 <= LD1t; clk1 <= clk1t;
 clkn1 <= clkn1t; clear1 <= clear1t;
END Structure;
--------------------------------------------------------------------
library altera;
use altera.altera_primitives_components.all;
LIBRARY ieee;
USE ieee.std_logic_1164.all;
Use Ieee.std_logic_unsigned.all;

ENTITY JR1_clocks Is
            PORT ( clkin_board,reset: IN STD_LOGIC;
          A2Dout: IN BIT;
          TXD: OUT BIT;
```

```
          clk,clkn, Phi1, Phi2, RS, clear,LD,A2Dclk,A2DCS,clk1,clkn1
          ,clear1,LD1: OUT STD_LOGIC );
END JR1_clocks;
ARCHITECTURE Structure OF JR1_clocks IS
COMPONENT Clkdivider
PORT ( clkin, resetn: IN STD_LOGIC;
       clkout: OUT STD_LOGIC_VECTOR (7 downto 0));
END COMPONENT;
component serialport_bit is
        port
          ( clkin,send_data: in std_logic;
               data_in: in bit_vector (7 downto 0);
               TXD: out bit);
End component;
Signal clktemp,cleartemp,LDtemp,RSt,RStNew,resetn,clkout1,clear1temp,
LD1temp,clk1temp: STD_LOGIC;
Signal Send_data,readtest,loadtest,clearUp: STD_LOGIC :='0';
Signal reset2: STD_LOGIC:= '1';
Signal A2DCSt: STD_LOGIC:='1';
Signal clktemp3,clktemp4,clktemp2,clkin, clkin1,clkin2,clktempV,
clkintemp,clkM: STD_LOGIC_VECTOR (7 downto 0);
Signal Digvec,temp: bit_vector(23 downto 0);
Signal Data_in: bit_vector(7 downto 0);
signal acc: std_logic_vector (15 downto 0);
signal RSKc: std_logic;
signal RSKc2: std_logic;
```

```
shared variable n: integer range 0 to 511:=0 ;
shared variable i,j,k,ii,jj: integer range 0 to 63 :=0 ;
shared variable res: integer range 0 to 256:=0;
shared variable CE: integer range 0 to 1:=0;
shared variable clearC,RSk: integer range 0 to 7:=0;
shared variable clkC: integer range 0 to 31:=0;
type aray is array (255 downto 0) of bit_vector (23 downto 0);
Signal Digarray: aray;
Constant M: integer:=34;
BEGIN
process (reset)
begin
   if n=257 then
      reset2<='0';
   elsif falling_Edge (reset) then
      reset2<='1';
  end if;
end process;
process (clktemp)
begin
IF reset='0' then
    res:=0;
Else
  IF  Rising_Edge(clktemp)  THEN
    if j=0 then
       res:=0;
```

```
     Else
       res:=res+1;
    end if;
  End if;
End if;
end process;
resetn <= reset;
clockdivider0: Clkdivider PORT MAP(clkin_board,resetn,clkin1);
clockdivider1: Clkdivider PORT MAP(clkin1(7),resetn,clkin);
clockdivider2: Clkdivider PORT MAP(clkin(7),resetn,clktempV);
clktemp <= clktempV(3);clk <= clktemp;clkn <= NOT clktemp;
Phi1 <= clkin(7);Phi2 <=  NOT clkin(7);
Process (clkin(7), clktemp)
variable  RSCounter: integer range 0 to 10:=0;
variable  RSCS: integer range 0 to 1:=0;
Begin
if resetn='0' then
   RSCounter:=0;
else
IF Falling_Edge(clktemp) then
  RSCS:=1;
end if;
 if RSCounter=8 then
    RSCS:=0;RSt<='1';
 else
   RSt<='0';
```

```
end if;
IF  Falling_Edge(clkin(7)) then
if RSCounter<8 and RSCS=1 then
    RSCounter:=RSCounter+1;
    elsif RSCS=0 then
    RSCounter:=0;
end if;
end if;
end if;
end process;
RStNew <= RSt and (NOT clkin(7)) ;
RS <= RSt;
Process (RSt)
Begin
IF  Rising_Edge(RSt) then
    RSkc<='1';
end if;
if RSk>1 then
   RSkc<='0';
 end if;
end process;
Process (clkin1(2))
Begin
IF  Rising_Edge(clkin1(2)) then
if RSKc='1' then
 RSk:=RSk+1;
```

```
 else
 RSk:=0;
end if;
end if;
end process;
Process (cleartemp)
Begin
 If reset='0'then
     clearC:=0;
 Else
    IF Rising_Edge (clktemp) then
       IF clearC<6  then
          clearC:=clearC+1;
       end if;
    end if;
 end if;
    If cleartemp='1' then
       clearUp<='1';
    else
       clearUp<='0';
    end if;
end process;
Process (clktemp,clkin(7))
 Begin
if resetn='0' or res=M then
    i:=0;j:=0;k:=0; CE:=0;cleartemp <='0';LDtemp <='0';
```

```
else
  IF  Rising_Edge(clkin(7)) then
        if CE=1 and k<62 THEN
          k:=k+1;
        elsif CE=0 then
          k:=0;
        end if;
  end if;
    IF  Falling_Edge(clktemp) and j>0  THEN
        if i<63  then
         i := (i + 1);
        end if;
     END IF;
     IF  Rising_Edge(clktemp)  THEN
         if j<63  then
         j := (j + 1);
         end if;
     END IF;
 end if;
 if (i=2 and j=2) or (i=3 and j=3) then
    CE:=1;
 Else
    CE:=0;
 end if;
  if i=0 and j=0 then
     cleartemp <='0';LDtemp <='0';
```

```
  elsif j<3 and j>0 and (i=0 or i=1) then
     cleartemp <='1';LDtemp <= '0';
  elsif i=2 and j=2  then
          if k>3 then
            LDtemp <= '1';cleartemp <='0';
          else
             LDtemp <= '0';cleartemp <='0';
          end if;
  elsif i=2  and j=3 then
        LDtemp <= '1'; cleartemp <='0';
  elsif i=3 and j=3 then
          if k<3 then
             LDtemp <= '1';cleartemp <='0';
          else
             LDtemp <= '0';cleartemp <='0';
          end if;
    elsif i>3 and j>3 then
   LDtemp <= '0';cleartemp <='0';
   end if;
END Process;
clear<=cleartemp;LD <= LDtemp;
Process (clktemp,cleartemp)
 variable clk1C: integer range 0 to 31:=0;
 variable clk1CE: integer range 0 to 1;
 Begin
 IF Rising_Edge(cleartemp) then
```

```
  if clk1C =0 then
   clk1CE:=1;
  end if;
  end if;
  if clk1C =10 then
   clk1CE:=0;
  end if;
 IF Rising_Edge(clktemp) then
   if clk1CE=0 then
      clk1C := 0;clk1temp <='0';
   elsif clk1CE=1 then
      clk1C := clk1C+1; clk1temp <='1';
    end if;
end if;
 END Process;
clk1 <= clktemp;clkn1 <= NOT clktemp;
A2Dclk <= NOT clkin(4);
Process (cleartemp,resetn)
 Begin
if resetn='0' then
     ii:=0;jj:=0; LD1temp <= '0'; clear1temp <= '0';
elsif resetn='1' then
     IF  Rising_Edge(cleartemp) THEN
        if ii<7  then
         ii := (ii + 1);
        end if;
```

```
      END IF;
      IF  Falling_Edge(cleartemp) and ii>0  THEN
          if jj<7  then
          jj := (jj + 1);
         end if;
      END IF;
      if ii>0 and jj<2 then
         clear1temp <= '1';
      else
         clear1temp <= '0';
      end if;
        if ii>1 and jj>1 and jj<3 then
         LD1temp <= '1';
      else
          LD1temp <= '0';
      end if;
end if;
end process;
clear1<=cleartemp;LD1<=LDtemp;
process(clkin(7),RSt, clkin(4))
variable RSC: integer range 0 to 16 :=0;
variable CSC,phi1C:  integer range 0 to 31 :=0;
variable phi1C_start: std_logic:='0';
variable A2Dclk_start: integer range 0 to 1;
variable Conv_done: integer range 0 to 1 :=0;
variable Conv_C: integer range 0 to 31 :=0;
```

```
begin
if resetn='0' then
       RSC:=0; phi1C:=0; Conv_done:=0; A2Dclk_start:=0;
       CSC:=0;phi1C_start:='0';A2DCSt<='1';Conv_C:=0;
else
IF Rising_Edge(RSt) THEN
 phi1C_start:='1';
     if RSC<3 then
       RSC:=RSC+1;
     end if;
 end if;
  if phi1C=16 then
       phi1C_start:='0';
 end if;
IF Falling_Edge(clkin(7)) THEN
       if phi1C<16 and phi1C_start='1' then
        phi1C:=phi1C+1;
        end if;
        if phi1C=16 then
          phi1C:=0;
        end if;
         if phi1C=13 then
            A2Dclk_start:=1;
          end if;
         if Conv_done=1 then
            A2Dclk_start:=0;
```

```
          end if;
end if;
IF  Rising_Edge(clkin(4)) THEN
         if A2Dclk_start=1 and CSC<30  then
        CSC:=CSC+1;
          A2DCSt <= '0';
    end if;
       if CSC=29 then
          Conv_done:=1;
       end if;
      if CSC>29 then
          A2DCSt <= '1';
      end if;
       if A2Dclk_start=0 then
          CSC:=0;Conv_done:=0;
       end if;
end if;
end if;
end process;
A2DCS<=A2DCSt or cleartemp;
Process(A2DCSt, clkin(4),resetn)
variable m: integer range 0 to 63:=0 ;
begin
 if resetn='0' then
  n:=0;m:=0;
elsif resetn='1' and clearC>3 then
```

```
  if falling_Edge (A2DCSt) then
    n:=n+1;
 end if;
 IF falling_Edge( clkin(4)) THEN
    if A2DCSt='0' and clearUp='0' then
      m:=m+1;
     elsif A2DCSt='1' or clearUp='1' then
      m:=0;
    end if;
    if m>3 and m<27 then
       Digvec(m-4)<=A2Dout;
       readtest <= NOT readtest;
    elsif m=27 then
       Digvec(m-4)<= '0';
        loadtest <= NOT loadtest;
     elsif m=28 then
       if n<257 then
         Digarray(n-1)<=Digvec;
       end if;
    end if;
 end if;
end if;
end process;
Process (clkin_board)
begin
 if  Rising_Edge(clkin_board) then
```

```
                acc <= acc+ 151; clkout1 <= acc(15);
 end if;
end process;
serialport1: serialport_bit PORT MAP (clkout1,Send_data,Data_in,TXD);
clockdivider11: Clkdivider PORT MAP(clkout1,'1',clkM);
Process (clkout1,clkM(5))
variable v,t:integer range 0 to 511:=0;
variable u:integer range 0 to 1:=1;
variable done:integer range 0 to 1:=0;
begin
if reset2='1' then
v:=0;u:=1;t:=0;done:=0;
elsif reset2='0' then
if rising_edge (clkM(5)) and done=0 then
   if v<256 then
     v:=v+1;temp<= Digarray(v-1);u:=0;
    else
     done:=1;v:=0;
   end if;
end if;
if rising_edge (clkout1) and u=0  and done=0 then
 t:=t+1;
  if t>0 and t<15 then
       Data_in <= temp(7 downto 0);
       Send_data<='1';
    elsif t=16 then
```

```
        Send_data<='0';
    elsif t>16 and t<31 then
        Data_in <= temp(15 downto 8);
        Send_data<='1';
    elsif t=32 then
        Send_data<='0';
    elsif t>32 and t<47 then
        Data_in <= temp(23 downto 16);
        Send_data<='1';
    elsif t>47 and t<63 then
        Send_data<='0';
    elsif t>63 then
          t:=0;
 end if;
 end if;
end if;
end process;
END Structure;
--------------------------------------------------------------------
-- Clock devider:
library altera;
use altera.altera_primitives_components.all;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Clkdivider IS
```

```
   PORT ( clkin, resetn: IN STD_LOGIC;
          clkout: OUT STD_LOGIC_VECTOR (7 downto 0));
END Clkdivider;

ARCHITECTURE Structure OF Clkdivider IS
Signal Qs, ts: STD_LOGIC_VECTOR (7 downto 0);
 Signal clkint: STD_LOGIC;

BEGIN
 clkint <= clkin and resetn;
 C0: JKFF PORT MAP ('1','1', clkint,'1','1', Qs(0));
C1: JKFF PORT MAP (Qs(0),Qs(0), clkint,'1','1',Qs(1));
ts(1) <= (Qs(1)AND Qs(0));
C2: JKFF PORT MAP (ts(1),ts(1), clkint,'1','1',Qs(2));
ts(2)<= (ts(1) AND Qs(2));
C3: JKFF PORT MAP (ts(2),ts(2), clkint,'1','1',Qs(3));
ts(3)<= (ts(2) AND Qs(3));
C4: JKFF PORT MAP (ts(3),ts(3), clkint,'1','1',Qs(4));
ts(4)<= (ts(3) AND Qs(4));
C5: JKFF PORT MAP (ts(4),ts(4), clkint,'1','1',Qs(5));
ts(5)<= (ts(4) AND Qs(5));
C6: JKFF PORT MAP (ts(5),ts(5), clkint,'1','1',Qs(6));
ts(6)<= (ts(5) AND Qs(6));
C7: JKFF PORT MAP (ts(6),ts(6), clkint,'1','1',Qs(7));
clkout <= Qs;
END Structure;
```

```
------------------------------------------------------------------
-- To provide the serial port connection:
LIBRARY ieee;
         use IEEE.std_logic_1164.all;use IEEE.std_logic_arith.all;
        use IEEE.numeric_bit.all;use IEEE.numeric_std.all;
        use IEEE.std_logic_signed.all;use IEEE.std_logic_unsigned.all;
        use IEEE.math_real.all; use IEEE.math_complex.all;

     Entity serialport_bit is
          port
          (    clkin,send_data: in std_logic;
               data_in: in bit_vector (7 downto 0);
               TXD: out bit);
     End serialport_bit;
ARCHITECTURE Structure OF serialport_bit IS
          signal TXDVector: bit_VECTOR (9 downto 0);
          shared variable i: integer range 0 to 15 ;
          signal TXDtemp: bit := '1';
          begin
          TXDVector ( 8 downto 1) <= data_in;
          TXDVector(0) <= '0';TXDVector(9) <= '1';
          process (clkin, send_data)
               begin
               if  (send_data='0') then
               i:=0;TXDtemp<='1';
               elsif  Rising_Edge(clkin) and (i<10)  then
```

```
                TXDtemp <=  TXDVector(i);i:=i+1;
                end if;
           end process;
                TXD <= TXDtemp;
End Structure;
```

VHDL codes for the experimental set-up (video coding measurements):

```
library altera;
use altera.altera_primitives_components.all;
LIBRARY ieee;
USE ieee.std_logic_1164.all;
Use Ieee.std_logic_unsigned.all;

-- To provide:
-- 1. Row and column select signals:
--    A, and B.
-- 2. Enable signal for the pixel:
--    En.
-- 3. Reset signal for the pixel:
--    RS.
ENTITY ControlSignals IS
   PORT ( clkin_board,reset: IN STD_LOGIC;
          A,B,RS,En: OUT STD_LOGIC);
END ControlSignals;
ARCHITECTURE Structure OF ControlSignals IS
```

```
COMPONENT Clkdivider
PORT ( clkin, resetn: IN STD_LOGIC;
       clkout: OUT STD_LOGIC_VECTOR (7 downto 0));
END COMPONENT;
Signal clkAB,clktemp,Btemp,Atemp,RSt,resetn: STD_LOGIC;
Signal Avec, Bvec, clkin1,clktempV,clkin: STD_LOGIC_VECTOR (7 downto 0);
shared variable i: integer range 0 to 7 :=0 ;
shared variable j: integer range 0 to 1 :=0 ;
BEGIN
resetn <= reset;
clockdivider0: Clkdivider PORT MAP(clkin_board,resetn,clkin1);
clockdivider1: Clkdivider PORT MAP(clkin1(7),resetn,clkin);
clockdivider2: Clkdivider PORT MAP(clkin(7),resetn,clktempV);
clktemp <= clktempV(4);
Avec<="11111111";
Bvec<="11010001";
clkAB <= clktempV(2);
process (clkAB)
begin
IF  Rising_Edge(clkAB)  THEN
  A<= Avec(i);B<= Bvec(i);j:=1;
elsif falling_edge(clkAB) and j=1 then
i:=i+1;
end if;
end process;
D0: DFF PORT MAP ('1',clktemp, clkin(7), '1', RSt);
```

```
RS <= RSt;En <='1';
END Structure;
```