DOCTORAL THESIS

ON THE DESIGN OF COMPRESSED SENSING CMOS IMAGERS

AUTHOR:

MARCO TREVISI

ADVISORS:

RICARDO CARMONA GALÁN

ÁNGEL RODRÍGUEZ VÁZQUEZ

SEVILLE 2021

ON THE DESIGN OF COMPRESSED SENSING CMOS IMAGERS

BY:

MARCO TREVISI

PROPUESTA DE TESIS DOCTORAL

PARA LA OBTENCIÓN DEL GRADO DE

DOCTOR EN CIENCIAS Y TECNOLOGÍAS FÍSICAS

ADVISORS:

RICARDO CARMONA GALÁN

ÁNGEL RODRÍGUEZ VÁZQUEZ

SEVILLE 2021

THE DESIGN OF COMPRESSED SENSING CMOS IMAGERS

DOCTORAL THESIS
UNIVERSIDAD DE SEVILLA

MARCO TREVISI
MASTER IN AERONAUTICS

ADVISOR: RICARDO CARMONA GALÁN
TENURED SCIENTIST
INSTITUTO DEMICROELECTRONICA DE SEVILLA
CONSEJO SUPERIOR DE INVESTIGACIONES

ADVISOR: ÁNGEL RODRÍGUEZ VÁZQUEZ
FULL PROFESSOR
ELECTRONICS AND ELECTROMAGNETISM DEPARTMENT
UNIVERSIDAD DE SEVILLA

EXTERNAL REVIEWER: WILLIAM GUICQUERO
SENIOR RESEARCHER
L3I SMART EMBEDDED IMAGING SYSTEMS LAB
CEA-LETI GRENOBLE

EXTERNAL REVIEWER: PÉTER FÖLDESY
SENIOR RESEARCHER
INSTITUTE FOR COMPUTER SCIENCE AND CONTROL (MTA-SZTAKI)
HUNGARIAN ACADEMY OF SCIENCES (MTA)

UNIVERSIDAD DE SEVILLA
CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS
CENTRO NACIONAL DE MICROELECTRÓNICA
INSTITUTO DE MICROELECTRÓNICA DE SEVILLA

CALLE AMÉRICO VESPUCIO, 28.
PARQUE CIENTÍFICO Y TECNOLÓGICO CARTUJA.
41092 SEVILLA
AUTHOR E-MAIL: MARCO.TREVISI@HOTMAIL.COM

A Lidia, por haber sido
siempre nosotros

Per Aspera sic itur
ad Astra

ABSTRACT

Compressive Sampling (CS) is a sampling theory and an alternative to the sampling process based on the Nyquist–Shannon's theorem [Cand06]. While conventional sampling applies the Whittaker–Shannon interpolation formula [Whit15] to recover a continuous-time signal from a discrete set of time samples, CS replaces it with recovery by means of convex optimization of its L_1 norm [Bara07]. In doing so CS transforms the conventional limit imposed on the high frequency components of a continuous signal that undergoes sampling into a limit imposed on its sparseness i.e. on the number of relevant elements that it contains.

This sampling paradigm has given rise to new sensors that generate what, in literature, have become known as compressed samples. These compressed samples, needed by CS reconstruction algorithms to recover a signal, are linear combinations of point values of the sampled signal weighted by random coefficients. These coefficients, joined together, form a measurement matrix. Just like standard converters have to abide to a certain sampling rate to avoid aliasing, a measurement matrix needs to respect the Restricted Isometry Property (RIP) to generate viable compressed samples.

CMOS Image sensors that are designed applying this new theory (CS-CIS) need to incorporate a measurement matrix in their architecture. Some designers add it using optical elements in front of the pixel array while others insert dedicated circuitry on-chip. Those that choose this second approach include in their prototypes binary pseudo-random number generators (PRNG) whose outputs are used to randomly select the pixels of the array. CS optimal measurement matrix is the orthonormalised Gaussian matrix, a matrix in which each element is extracted at random from a normalised Gaussian distribution that then undergoes an orthogonalisation process. PRGN used in the design of CS-CIS generate binary measurement matrices that are not orthonormal and that are random only in appearance. This fact has a direct consequence on the quality of the compressed samples that these CS-CIS deliver: the RIP of these non-ideal matrices is limited by a low sparsity constant and thus greatly limits the number of relevant elements that a sampled image can have.

In this work we have explored the RIP of random binary measurement matrices to quantify

this limitation. We also have analysed the behaviour of different types of PRNG using mathematical tools such as the power spectral density applied to pattern recognition to see how much these elements approach the performance of truly random binary matrices. We have shown that the dynamic behaviour of class III elementary cellular automata (ECA) implementing rule 30 makes them the best PRNG choice for on-chip recursive measurement matrix generation and as such they produce the best compressed samples that a matrix of this characteristics can deliver.

Besides studying the quality of the compressed samples from a theoretical perspective we have also analysed their quality from an electrical point of view. Each compressed sample that a CS-CIS generates is created by summing together pixel contributions in the analog domain. In its original form an image is not sparse, for this reason the dynamic range needed to properly describe a linear combination of its point values is quite high and the number of bits needed to digitise a compressed sample far surpasses that of a single pixel. We have defined this number and we have studied the limits of A/D conversion to see if proper digitisation of compressed samples using an analog-to-digital converter ADC was feasible. As a viable alternative we have proposed pixel pulse modulation schemes that transpose the lack of dynamic range into a problem of time consumption thus increasing the amount of bits at our disposal to describe a compressed sample.

We have joined the results of these two studies in a CS-CIS of our own design. It has a 64×64 pixel array; it implements a rule-30 ECA for on-chip measurement matrix generation and a pulse width modulation of its pixels to deliver 20-bit compressed samples. A prototype of this imager has been built using a CMOS 0.18μm 1P6M technology. We have presented the results of a set of experiments carried out on this prototype to test its performance and confirm that pulse width modulation of 8-bit pixels is indeed a feasible solution to create a CS-CIS capable of delivering a stream of 20-bit compressed samples at 30fps with a minimum compression ratio of 0.4 in order to take into account the different size between pixels and compressed samples.

We have found a new way to manipulate the output of a CS-CIS to achieve motion detection using a non-recursive algorithm without resorting to image recovery or CS background subtraction. We have studied its performance by means of Root-Mean-Square Error (RMSE) using MATLAB. While not as precise as other solutions our results show that it can be used in real time because it does not need the large computational burden that CS reconstruction algorithms require.

Analysing how to increase the performance of CS-CIS measurement matrices from a theoretical perspective, we have proposed a new CS-CIS architecture that uses PRNG and a

differential readout system for the pixel array that combined generate pseudo-random ternary measurement matrices. Using Peak Signal-to-Noise Ratio (PSNR) as an image quality metric, through MATLAB simulations, we have shown that it outperforms binary matrices almost reaching reconstruction results similar to an ideal Gaussian matrix.

Lastly we have studied a way to transform the Harris corner detection algorithm into a sparsifying dictionary. The quasi-linearity of its equations and the fact that its output is given in the form of a sparse matrix with the same size of the pixel array having relevant coefficients only in correspondence of a corner makes it the ideal candidate for the creation of a ready-to-use sparsifying dictionary. Such a dictionary does not represent a sparsifying basis for the whole image; it is used to prune the compressed samples and preserve only information about the corners. This is ideal for compressed samples delivered by CS-CIS that suffer from the low sparsity constant of the matrix used to take them. We have tested its efficiency comparing corner extraction from compressed samples achieved using this dictionary on a reconstruction algorithm with corner detection on the image reconstructed using these same samples and the same algorithm. As parameters for comparison we have used the number of false positives, the number of false negatives and the distance of the corners from the ground truth. In all of them, corner extraction outperformed corner detection on the reconstructed image.

RESUMEN

El muestreo compresivo (CS) es una teoría de muestreo y una alternativa al proceso de muestreo basado en el teorema de Nyquist-Shannon [Cand06]. Mientras que el muestreo convencional aplica la fórmula de interpolación de Whittaker-Shannon [Whit15] para recuperar una señal temporal continua a partir de un conjunto discreto de muestras temporales, CS la reemplaza con su reconstrucción mediante optimización convexa de su norma L_1 [Bara07]. Al hacerlo, CS transforma el límite convencional impuesto sobre los componentes de alta frecuencia de una señal continua que se somete a muestreo en un límite impuesto sobre su dispersión, es decir, sobre el número de elementos relevantes que contiene.

Este paradigma de muestreo ha dado lugar a nuevos sensores que generan lo que, en la literatura, se conocen como muestras comprimidas. Estas muestras comprimidas, necesarias para que los algoritmos de reconstrucción basados en CS puedan funcionar, son combinaciones lineales de valores puntuales de la señal muestreada ponderados por coeficientes aleatorios. Estos coeficientes, unidos entre sí, forman una matriz de medición. Al igual que los convertidores estándar tienen que cumplir con una determinada frecuencia de muestreo para evitar el aliasing, una matriz de medición debe respetar la propiedad de isometría restringida (RIP) para generar muestras comprimidas viables.

Los sensores de imagen CMOS que se diseñan a partir de esta nueva teoría (CS-CIS) necesitan incorporar una matriz de medición en su arquitectura. Algunos diseñadores la añaden utilizando elementos ópticos delante de los píxeles, mientras que otros insertan circuitos dedicados on-chip. Los que eligen este segundo enfoque incluyen en sus prototipos generadores de números pseudoaleatorios binarios (PRNG) cuyas salidas se utilizan para seleccionar aleatoriamente los píxeles del sensor. Una matriz de medición óptima en CS es la matriz gaussiana ortonormalizada, una matriz en la que cada elemento se extrae al azar de una distribución gaussiana normalizada y que luego se somete a un proceso de ortogonalización. PRGN utilizados en el diseño de CS-CIS generan matrices de medición binarias que no son ortonormales y que son aleatorias solo en apariencia. Este hecho tiene una consecuencia directa en la calidad de las muestras comprimidas que estos CS-CIS recogen: la RIP de estas matrices no

ideales está limitada por una constante de dispersión baja y, por lo tanto, limitan mucho el número de elementos relevantes que una imagen muestreada puede tener.

En este trabajo hemos explorado la RIP de matrices de medición binarias aleatorias para cuantificar esta limitación. También hemos analizado el comportamiento de diferentes tipos de PRNG utilizando herramientas matemáticas como la densidad espectral de potencia aplicada al reconocimiento de patrones para ver cuánto se acercan dichos PRNG al rendimiento de matrices binarias verdaderamente aleatorias. Hemos demostrado que el comportamiento dinámico de los autómatas celulares elementales de clase III (ECA) que implementan la regla 30 los convierte en la mejor opción para la generación de matrices de medición recursivas y, como tal, producen las mejores muestras comprimidas que una matriz de estas características puede proporcionar.

Además de estudiar la calidad de las muestras comprimidas desde una perspectiva teórica, también la hemos analizado desde un punto de vista eléctrico. Cada muestra comprimida que genera un CS-CIS se crea sumando contribuciones de píxeles en el dominio analógico. En su forma original una imagen no es dispersa y, por esta razón, el rango dinámico necesario para describir adecuadamente una combinación lineal de sus valores puntuales es bastante alto, por consecuencia el número de bits necesarios para digitalizar una muestra comprimida supera con creces el necesario para digitalizar un solo píxel. Hemos definido este número y hemos estudiado los límites de la conversión A/D para ver si una digitalización adecuada de muestras comprimidas era factible utilizando un convertidor de señal analógica a digital (ADC). Como alternativa viable al uso de un ADC, hemos propuesto esquemas de modulación asíncrona de los píxeles por ancho o por frecuencia de pulso para transponer la falta de rango dinámico disponible para la representación de una muestra comprimida en el dominio analógico en un problema de consumo de tiempo durante el barrido de los píxeles, aumentando así la cantidad de bits a nuestra disposición para describir dicha muestra comprimida.

Hemos juntado los resultados de estos dos estudios para diseñar nuestro prototipo de CS-CIS. Tiene una matriz de 64×64 píxeles, implementa un ECA que evoluciona siguiendo la regla 30 para la generación de una matriz de medición on-chip y una modulación asíncrona por ancho de pulso de sus píxeles para recoger muestras comprimidas de 20 bits. Hemos diseñado este prototipo utilizando una tecnología CMOS 0.18μm 1P6M. Presentamos los resultados de un conjunto de experimentos realizados para comprobar su rendimiento y confirmar que la modulación de ancho de pulso de pixeles de 8 bits es una solución eficaz: nuestro CS-CIS es capaz de entregar un flujo de muestras comprimidas de 20 bits a 30 fps con una razón de compresión mínima de 0.4 para tener en cuenta la diferencia de tamaño entre píxeles y muestras comprimidas.

También hemos estudiado una nueva forma de manipular las salidas de un CS-CIS para lograr la detección de movimiento con un método no recursivo y sin recurrir al uso de algoritmos de reconstrucción. Hemos estudiado el rendimiento de este procedimiento mediante la raíz del error cuadrático medio (RMSE) en simulaciones de MATLAB. Si bien nuestro método no es tan preciso como otros, los resultados muestran que se puede usar en tiempo real porque no necesita la gran carga computacional que requieren los algoritmos de reconstrucción clásicos.

Analizando cómo aumentar el rendimiento de las matrices de medición empleadas en el diseño de CS-CIS desde una perspectiva teórica, hemos propuesto una nueva arquitectura que utiliza PRNG y un sistema de lectura diferencial de los píxeles que combinados generan matrices de medición ternarias pseudoaleatorias. Utilizando la proporción máxima de señal a ruido (PSNR) como una métrica de calidad de imagen, a través de simulaciones MATLAB, hemos demostrado que estas matrices ternarias superan el rendimiento de las matrices binarias casi alcanzando resultados de reconstrucción similares a una matriz gaussiana ideal.

Por último, hemos estudiado una forma de transformar el detector de esquinas de Harris en un diccionario dispersivo. Hemos elegido este algoritmo por la cuasi-linealidad de sus ecuaciones y por el hecho de que su salida se da en forma de matriz dispersa, cuyo tamaño es igual al de la imagen elaborada y cuyos coeficientes son relevantes solo en correspondencia de posibles esquinas. El diccionario obtenido de este modo no representa una transformación de base para toda la imagen; más bien se puede usar para filtrar un conjunto de muestras comprimidas y preservar solo la información sobre esquinas que dichas muestras contienen. La aplicación de este diccionario es interesante especialmente en el caso de muestras comprimidas generadas por CS-CIS, ya que su precisión se ve reducida por la baja constante de dispersión de las matrices de medición que los CS-CIS incorporan. Hemos estudiado la eficacia de este método confrontando la extracción de esquinas obtenidas a partir de un algoritmo de reconstrucción que incorpore nuestro diccionario con la detección de esquinas obtenidas usando el detector de Harris original sobre la imagen tras su reconstrucción. Como parámetros de comparación, hemos utilizado el número de falsos positivos, el número de falsos negativos y la distancia que las esquinas detectadas tienen desde su posición en la imagen original. En todos los casos, la extracción de esquinas obtenidas usando el diccionario de dispersión superó la detección de esquinas obtenidas usando el algoritmo original.

RIASSUNTO

Compressive Sampling (CS) è una teoria di campionamento e un'alternativa al processo basato sul teorema di Nyquist-Shannon [Cand06]. Mentre il campionamento convenzionale applica la formula di interpolazione di Whittaker-Shannon [Whit15] per recuperare un segnale continuo nel tempo da una discreta serie di campioni temporali, CS lo sostituisce con il recupero mediante l'ottimizzazione convessa della sua norma L_1 [Bara07]. In tal modo CS trasforma il limite convenzionale imposto sui componenti d'alta frequenza del segnale continuo che si campiona in un limite imposto sulla sua sparsità, cioè sul numero di elementi rilevanti che contiene.

Questo paradigma di campionamento ha dato origine a nuovi sensori che generano quelli che, in letteratura, sono sono noti come campioni compressi. Questi campioni compressi, necessari agli algoritmi di ricostruzione CS per recuperare un segnale, sono combinazioni lineari di valori puntuali del suddetto segnale ponderati da coefficienti aleatori. Questi coefficienti, uniti insieme, formano una matrice di misurazione. Proprio come i convertitori standard devono rispettare una certa frequenza di campionamento per evitare l'aliasing, una matrice di misurazione deve rispettare la proprietà di isometria ristretta (RIP) per generare campioni compressi funzionali.

I sensori di immagine CMOS progettati a partire questa nuova teoria (CS-CIS) devono incorporare una matrice di misurazione nella loro architettura. Alcuni designer la aggiungono utilizzando elementi ottici davanti ai pixel, mentre altri la inseriscono usando circuiti dedicati on-chip. Quelli che scelgono questo secondo approccio includono, nei loro prototipi, dei generatori di numeri binari pseudo-aleatori (PRNG) le cui uscite sono usate per selezionare a caso i pixel dell'array. Una matrice di misurazione ottimale per il CS è la matrice gaussiana ortonormalizzata, una matrice in cui ogni elemento viene estratto a caso da una distribuzione di probabilità gaussiana normalizzata che poi subisce un processo di ortogonalizzazione. I PRGN utilizzati nella progettazione di CS-CIS generano matrici binarie che non sono ortonormali e che sono aleatorie solo in apparenza. Questo fatto produce conseguenze dirette sulla qualità dei campioni compressi che questi CS-CIS catturano: la PIR di queste matrici non ideali è limitata da una costante di sparsità bassa che limita notevolmente il numero di elementi rilevanti che

un'immagine campionata può avere.

In questo lavoro abbiamo studiato la RIP di matrici di misurazione binarie aleatorie per quantificare questa limitazione. Abbiamo anche analizzato il comportamento di diversi tipi di PRNG usando strumenti matematici come la densità spettrale di potenza applicata al riconoscimento di schemi ripetitivi per vedere quanto, questi elementi, si avvicinino alle prestazioni di suddette matrici. Abbiamo dimostrato che il comportamento dinamico degli automi cellulari elementari di classe III (ECA) che implementano la regola 30 li rende la migliore scelta di PRNG per la generazione ricorsiva di matrici di misurazione on-chip e come tali producono i migliori campioni compressi che una matrice di queste caratteristiche possa fornire.

Oltre a studiare la qualità dei campioni compressi da una prospettiva teorica, abbiamo anche analizzato la loro qualità da un punto di vista elettrico. Ogni campione compresso generato da un CS-CIS viene creato sommando i contributi di vari pixel nel dominio analogico. Nella sua forma originale un'immagine non è sparsa e per questo motivo l'intervallo dinamico necessario per descrivere correttamente una combinazione lineare dei suoi valori puntuali è piuttosto elevato e, di conseguenza, il numero di bit necessari per digitalizzare un campione compresso supera di gran lunga quello necessario per digitalizzare un singolo pixel. Abbiamo definito questo numero formalmente e abbiamo studiato i limiti della conversione A/D per vedere se fosse possibile una corretta digitalizzazione di campioni compressi usando un Convertitore analogico-digitale (ADC). Come alternativa plausibile abbiamo proposto architetture di pixel basati su una modulazione di larghezza o di frequenza d'impulso per trasporre la mancanza di intervallo dinamico in un problema di consumo di tempo, aumentando così la quantità di bit a nostra disposizione per descrivere un campione compresso.

Abbiamo unito i risultati di questi due studi in un CS-CIS di nostra progettazione. Ha un array di 64×64 pixel, incorpora una ECA con regola 30 per la generazione di matrici di misurazione on-chip e una modulazione di larghezza d'impulso dei suoi pixel per fornire campioni compressi di 20 bit. Il prototipo di questo sensore d'immagini è stato realizzato utilizzando una tecnologia CMOS 0.18μm 1P6M. Abbiamo presentato i risultati di una serie di esperimenti condotti per testarne le prestazioni e confermare che la modulazione di larghezza d'impulso di pixels di 8 bits è una soluzione davvero efficace per creare un CS-CIS in grado di fornire un flusso di campioni compressi di 20 bit a 30 fps con un rapporto di compressione minimo di 0.4 per tenere in conto la diversa dimensione dei pixel e dei campioni compressi.

Abbiamo anche studiato un nuovo modo per manipolare l'output di un CS-CIS allo scopo di rilevare movimento utilizzando un algoritmo non ricorsivo senza applicare algoritmi di

ricostruzione convenzionali. Abbiamo studiato le sue prestazioni per mezzo del Root-Mean-Square Error (RMSE) con simulazioni in MATLAB. Sebbene il nostro metodo non sia preciso come altre soluzioni, i risultati dimostrano che può essere utilizzato in tempo reale perché non ha bisogno del grande carico computazionale richiesto dagli algoritmi di ricostruzione convenzionali.

Analizzando come aumentare le prestazioni delle matrici di misurazione dei CS-CIS da un punto di vista teorico, abbiamo proposto una nuova architettura di CS-CIS che utilizza PRNG e un sistema di lettura differenziale per l'array che, combinati, generano matrici di misurazione ternarie pseudo-aleatorie. Utilizzando il peak signal-to-noise ratio (PSNR) come metrica della qualità dell'immagine, sempre attraverso simulazioni MATLAB, abbiamo dimostrato che queste matrici superano le matrici binarie raggiungendo quasi risultati di ricostruzione similari a una matrice gaussiana ideale.

Infine, abbiamo studiato un modo per trasformare il rivelatore d'angoli di Harris in un dizionario sparsificante. Abbiamo scelto questo algoritmo per la quasi-linearità delle sue equazioni e per il fatto che il suo output è dato sotto forma di una matrice sparsa di dimensione uguale a quella dell'immagine da ricostruire e i cui coefficienti sono rilevanti solo in corrispondenza di possibili angoli. Il dizionario ottenuto in questo modo non rappresenta una trasformazione di base per l'intera immagine; piuttosto, può essere usato per filtrare una serie di campioni compressi e conservare solo le informazioni sugli angoli contenuti in quei campioni. L'applicazione di questo dizionario è di particolare interesse nel caso di campioni compressi generati da CS-CIS poiché la loro precisione è ridotta dalla bassa costante di sparsità delle matrici di misurazione adottate nella progettazione dei CS-CIS stessi. Abbiamo studiato l'efficacia di questo metodo confrontando l'estrazione degli angoli ottenuti da un algoritmo di ricostruzione che incorpora il nostro dizionario con il rilevamento degli angoli ottenuti utilizzando il rilevatore Harris sull'immagine originale dopo la sua ricostruzione. Come parametri di confronto, abbiamo utilizzato il numero di falsi positivi, il numero di falsi negativi e la distanza che gli angoli rilevati hanno dalla loro posizione nell'immagine originale. In tutti i casi, l'estrazione degli angoli ottenuti usando il dizionario sparsificante ha superato il rilevamento degli angoli ottenuti usando l'algoritmo di Harris originale.

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

A/D	ANALOG TO DIGITAL
AAF	ANTI-ALIASING FILTER
ADC	ANALOG TO DIGITAL CONVERTER
BCS	BLOCK-BASED COMPRESSIVE SAMPLING
CA	CELLULAR AUTOMATON
CIS	CMOS IMAGE SENSORS
CS	COMPRESSIVE SAMPLING
CS-CIS	CS CMOS IMAGE SENSOR
DFT	DISCRETE FOURIER TRANSFORM
ECA	ELEMENTARY CELLULAR AUTOMATA
ENOB	EQUIVALENT NUMBER OF BITS
FPGA	FIELD-PROGRAMMABLE GATE ARRAY
LFSR	LINEAR FEEDBACK SHIFT REGISTER
LPF	LOW-PASS FILTER
MRI	MAGNETIC RESONANCE IMAGING
OAAF	OPTICAL ANTI-ALIASING FILTER
OLPF	OPTICAL LOW-PASS FILTER
PCB	PRINTED CIRCUIT BOARD
PSNR	PEAK SIGNAL-TO-NOISE RATIO
PRNG	PSEUDO RANDOM NUMBER GENERATOR
PSD	POWER SPECTRAL DENSITY
PFM	PULSE FREQUENCY MODULATION
PWM	PULSE WIDTH MODULATION
RIP	RESTRICTED ISOMETRY PROPERTY
RMSE	ROOT MEAN SQUARE ERROR
SNR	SIGNAL TO NOISE RATIO

CHAPTER 1
INTRODUCTION

Compressive Sampling (CS) first appeared in 2006 within the field of signal processing [Cand06] as an alternative to the Nyquist–Shannon sampling theorem. The purpose of sampling is to capture the content of a continuous signal using a series of discrete measurements. The device in charge of producing this sequence of measurements, that in this process take the name of samples, is called sensor. This operation is feasible only given either prior knowledge or assumptions about the signal that we want to capture: the Nyquist–Shannon sampling theorem establishes a sufficient condition on the sample rate so that a discrete sequence of samples includes all the information from a continuous signal within a finite bandwidth. It states that, if the sample rate is twice the signal bandwidth, then the signal can be reconstructed perfectly by means of the Whittaker–Shannon interpolation formula [Whit15], Eq. (1).

$$s(t) = \sum_{i=-\infty}^{+\infty} \mathbf{s}(i\Delta T) \frac{\sin\left(\pi \frac{t-i\Delta T}{\Delta T}\right)}{\pi(t-i\Delta T)} \Delta T = \sum_{i=-\infty}^{+\infty} \mathbf{s}(i\Delta T)\, \mathrm{sinc}\left(\frac{t-i\Delta T}{\Delta T}\right) \quad (1)$$

where $\mathbf{s}(i\Delta T)$ is the sequence of discrete measurements, $s(t)$ is the reconstructed signal and ΔT is the sampling interval. For the reconstructed signal to be equal to the original, they must have a Fourier transform whose non-zero values are confined to the region $|f| \leq 1/2\Delta T$.

$f_s = 1/\Delta T$ is known as the sample rate and $f_s/2$ is the corresponding Nyquist frequency. If the high frequency components of the sampled signal exceeded the Nyquist frequency we would incur in a phenomenon called aliasing: copies of these high frequency components would be created within the region $|f| \leq 1/2\Delta T$, they would overlap with the already existing components of that region, adding their power to them and altering the sampled signal structure irredeemably. In this case $s(t)$, upon reconstruction by means of Eq. (1), would be different from the original. To solve this problem in practice the input signal is filtered using a low-pass filter (LPF) whose impulse response is $\mathrm{sinc}[(t - i\Delta T)/\Delta T]$ and whose input is the impulse train:

$$s(t) = \sum_{i=-\infty}^{+\infty} \mathbf{s}(i\Delta T)\delta(t - i\Delta T) \quad (2)$$

In other words, the frequency band of the signal that we want to sample is limited with a LPF whose cut-off frequency is $f_s/2$ and this filter takes the name of anti-aliasing filter (AAF). Its presence guarantees that the part of the original signal included in $|f| \leq 1/2\Delta T$ be sampled and reconstructed correctly.

For many applications, modern electric sensors are designed as mixed-signal systems that incorporate a certain level of pre-processing of the information and what they deliver are sequences of samples already in discrete domains. This is especially true for CMOS image sensors (Fig. 1) where their ease of use, low production cost, design flexibility and suitability for the production of on-chip cameras have relegated other types of image sensors to niche markets.

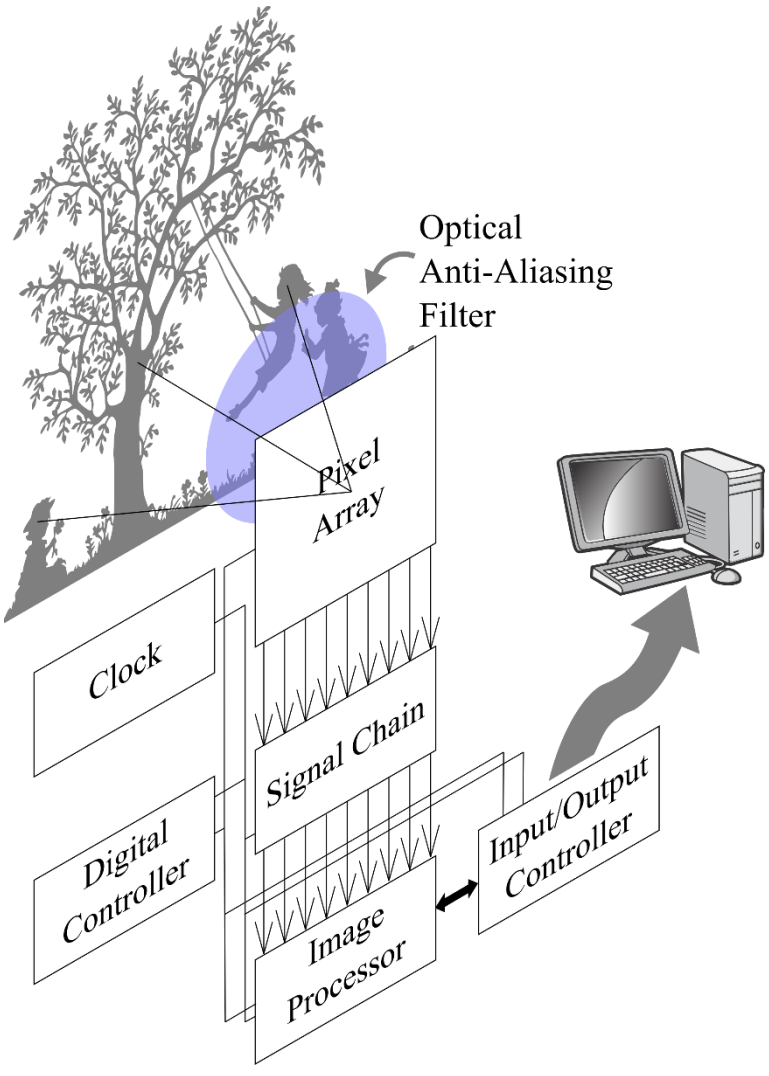

Fig. 1: Conventional digital image sensor diagram.

Generally speaking, the basic building blocks of CMOS image sensors are the pixels, discrete elements that contain a photodiode used to collect a spatial sample of a scene when the photons leaving that scene impact on its sensitive area. Each sample consists of a value that represents the

light intensity captured in a point of the array onto which the whole scene is reflected (Fig. 1). These values can then be described either by analog voltages or by analog currents.

Each photodiode is connected to three or more dedicated transistors (depending on the design and purpose) and the combination of each photodiode and its transistors is what forms a pixel. The number of transistor per pixel ranges from 3-4 transistors for standard digital cameras [Font13] up to few tens of transistors for smart imagers that introduce image pre-processing, correction of sampling errors or feature extraction at the focal plane [Foss97] [Fern16].

The analog-to-digital converter (ADC) reads the voltages from the pixels and converts them into digital outputs. Typical ADC architectures choices for image sensors are: dual-slope ADC, cyclic ADC, successive approximations ADC and Σ-Δ ADC [Leñe18]. Imagers implementing a serial readout employ a global ADC for the whole array, while column-parallel readout designs use one ADC for each imager column.

Most image processors are used to perform basic digital image processing in the digital domain, such as demosaicking, denoising and/or white-balancing. In some advanced architecture it is also used to extract features on the focal plane to perform object/face recognition or tracking [Fern16]. The I/O controller interfaces the image sensor with external electronics and, besides streaming image data, receives instructions used to set the internal registers of the imager that determine the operational mode and parameters. Finally, the digital controller manages the timed execution of the operations of the imager following the frequency of operation imposed by the clock.

In this representation of a conventional digital imager (Fig. 1) we can see that the sampling theorem imposes two fundamental limitations. Image sensors convert continuous optical scenes into discrete digital images containing information that varies across space and over time [Gonz92] [Seit00]. Temporal aliasing during image sampling is tied to the frame rate of the camera and in most cases it is not difficult to prevent. The temporal variations of a scene are synchronised with the imager's clock and happen at regular intervals of its period: since the pixels photo-currents of photo-charges are extremely weak unless the illumination levels are very high, pixel values are commonly obtained through integration and discrete-time operation processes where electrical charges get accumulated into capacitors over the clock period. This process can be effectively used to filter the signal over time and avoid temporal aliasing.

As for the variation of a scene across space note that we have described the pixel array as a series of discrete elements but, at the other side of the lenses, we find a natural environment comprised of spatially continuous elements. As such, we need to filter the incoming information in a space-wise fashion so that the discrete elements of the array do not incur spatial aliasing.

The extension of the Nyquist–Shannon sampling theorem to n-dimension takes the name of Petersen–Middleton theorem [Pete62]. The mathematics of two-dimensional spatial sampling is similar to that of time sampling and it delivers a similar result. The interpolation formula derived from this extension can be implemented in practice as an optical low-pass filter (OLPF) whose input is the impulse train:

$$\mathrm{x}(u,v) = \sum_{i,j=-\infty}^{+\infty} \mathbf{X}(i\Delta U, j\Delta V)\delta(u - i\Delta U, v - j\Delta V) \tag{3}$$

where $\mathbf{X}(i\Delta U, j\Delta V)$ is the matrix of discrete measurements of the image taken over a square lattice (i.e. pixel values), $x(u,v)$ is the reconstructed image and ΔU and ΔV are the spacing between pixels along the two dimensions of the array. The choice of OLPF, also known as blur filters or optical anti-aliasing filters (OAAF) (Fig. 1), involves a trade-off among sharpness, aliasing, and fill factor (i.e. the ratio of a pixel light sensitive area to its total area). The OLPF guarantees that the reconstructed image $\mathrm{x}(u,v)$ equals the original in a given range of spatial frequencies. In other words the sampling theorem establishes a direct dependence between the quality of a captured image and the pitch, number and placement of the pixels of the imager used to obtain it.

Upon sampling a signal, a common practice is that of applying compression algorithms. This is done because natural signals tend to be spatially piece-wise smooth (some well below the cut-off frequency imposed by the OAAF). The objective of these algorithms is to exploit this naturally occurring sparsity to reduce signal redundancy, minimising its bit size, in order to store or transmit it in an efficient form. Mathematically speaking, compression is achieved by transforming the domain in which a signal is represented. For digital images a wide range of sparse transforms can be used, some examples are the discrete Fourier transform, the discrete cosine transform (JPEG compression standard) [Ahme74] and the discrete wavelet transform (JPEG2000 compression standard) [Akan92]. When the signal to be compressed is represented in the discrete domain, such as the case of digital images, these transforms can take the form of linear combinations and be applied as matrix products:

$$\boldsymbol{\alpha} = f(\mathrm{X}) = \mathbf{T}\mathbf{x} \tag{4}$$

where the function $f(\cdot)$ represents the chosen transform, T is its discrete matrix representation, $\mathbf{x} \in \mathbb{R}^N$ is the matrix containing pixels values folded columnwise and $\boldsymbol{\alpha} \in \mathbb{R}^N$ is a set of parameters that represent the image in the transform domain chosen. Note that $\boldsymbol{\alpha}$ is still a vector that represents the sampled image but, in this new domain, this vector has a much lower entropy

than its counterpart **x** so compression can be achieved by pruning negligible elements of **α**. If we were to define M as the number of significant elements of **α** then $M \ll N$ so that mathematically speaking **α** can be considered a sparse representation of the original image.

CS sets itself as an alternative to the sampling and compression paradigm presented above by replacing signal recovery by means of interpolation Eq. (1) [Sten84] with signal recovery by means of convex optimisation [Trop10]:

$$min\|\mathbf{x}\|_{l_1} \text{ subject to } \|\mathbf{y} - \mathbf{\Phi x}\|_{l_2} < \varepsilon^2 \tag{5}$$

where **y** is a vector whose elements are defined as compressed samples, **x** is the vector of unknowns, **Φ** is called measurement matrix, ε is a scalar number that can be used to describe the measurement noise level of a zero mean additive white Gaussian noise and the operators $\|\cdot\|_{l_1}$ and $\|\cdot\|_{l_2}$ are p-norms:

$$\|\mathbf{x}\|_{l_p} = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p} \tag{6}$$

It is possible to notice that for $p = 1$ Eq. (6) represents the sum of the elements of **x** and for $p = 2$ it represents its Euclidean norm. Vector **x** contains the elements of the image that we want to reconstruct folded in a monodimensional array. In standard sampling, each one of these elements shares a biunivocal relationship with the actual output of a pixel within a sensor. In CS, as we will show with some examples in section 1.1, the two concepts are not tied in the same way. Moreover, since this sampling paradigm is not based on interpolation but on optimization, it is not affected by aliasing and pixel positioning (which are by-products of Eq. (3)). But, to simplify our argument without loss of generality, for the moment, let us consider the pixels in a sensor and the elements of an image (also known as pixels or picture elements) as one and the same. Using images as an example, given a pixel array of unknown values **x**, compressed samples **y** are obtained as linear combinations of the array elements [Bara07]:

$$\mathbf{y} = \mathbf{\Phi x} \tag{7}$$

From Eq. (7) we see that a measurement matrix **Φ** contains the set of parameters used to generate the compressed samples. CS differs from conventional techniques in that the original signal is not sampled at a fixed rate over regular time intervals Eq. (2) or over a square lattice Eq. (3) and subsequently compressed in a new domain to reduce its dimensionality Eq. (4); it is already sampled in a domain different from the discrete temporal or spatial domains of these equations. The compressed samples in $\mathbf{y} \in \mathbb{R}^M$ are elements that represent the image in a new

domain similarly to how the M significant elements of $\boldsymbol{\alpha} \in \mathbb{R}^N$ in Eq. (4) represented a signal in a transform domain. But the transformation in Eq. (7), despite taking the name of "compressive" due to its obvious similarities with Eq. (4), it is not a compression per se. What this sampling paradigm does is to exchange recovery by interpolation Eq. (1) with recovery by optimization Eq. (5).

The advantage of CS over standard sampling is that, depending on the orthonormality of $\boldsymbol{\Phi}$ [Bara07], it can operate as a space-wise compression algorithm in that, given $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{y} \in \mathbb{R}^M$, the information content of the sampled image can be stored using a number of compressed samples, M, lower than the number of pixels of the array N: although underdetermined problems, such as the inverse of Eq. (7) for $M < N$, are considered ill-posed because they lack a univocal solution, they can be solved if $\boldsymbol{\Phi}$ abides an extra constraint, the restricted isometry property (RIP):

$$(1 - \delta_k)\|\mathbf{x}\|_{l_2} \leq \|\boldsymbol{\Phi}_k \mathbf{x}\|_{l_2} \leq (1 + \delta_k)\|\mathbf{x}\|_{l_2} \tag{8}$$

being δ_k positive and arbitrarily small and $\|\boldsymbol{\Phi}_k^* \boldsymbol{\Phi}_k - \mathbf{I}\|_{l_2} \leq \delta_k$. RIP essentially requires that every set of rows of $\boldsymbol{\Phi}$ with cardinality less than k approximately behaves like an orthonormal system when operating on sparse vectors [Cand06]. Borrowing a mathematical statement [Bara10] we could say $\boldsymbol{\Phi}$ projects a higher dimensional space onto a lower dimensional space preserving Euclidean distances among represented elements. In simpler words, if $\boldsymbol{\Phi}$ respects the RIP and the image $\mathbf{x}$ that we want to sample is k-sparse, then Eq. (7) can be solved imposing a minimisation constraint on the elements of $\mathbf{x}$ as presented in Eq. (5). Convexity of Eq. (5) makes signal recovery by means of optimization feasible because any local minimum must also be a global minimum hence the uniqueness of the solution.

As such, the key requirement of CS is sparsity of the sampled signal and this property can either exist in the signal original domain or in another basis. This means that, were the sampled signal not sparse enough to be sampled properly with a given $\boldsymbol{\Phi}$, it is possible to multiply, a posteriori, the outcome of Eq. (7) by a transform matrix like those presented in Eq. (4) to lower the signal entropy, thus increasing its compressibility and be able to achieve correct reconstruction. In CS these transform matrices take the name of sparsifying dictionaries. Continuing with the example of digital images, as mentioned before, there is a wide range of transforms can be used to increase image sparsity (i.e. Fourier transform, discrete cosine transform or discrete wavelet transform). If we assume that an image is sparse in one of the aforementioned domains then Eq. (5) takes the form:

$$argmin\|\boldsymbol{\alpha}\|_{l_1} \text{ subject to } \|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha}\|_{l_2} < \varepsilon^2 \quad (9)$$

being $\boldsymbol{\Psi} = \mathbf{T}^{-1}$ (from Eq. (4)) the sparsifying dictionary, a matrix that, multiplied by a signal, returns $\boldsymbol{\alpha}$, the vector of coefficients of the transformation of that signal to the new sparse domain. In this case, if we introduce the matrix $\boldsymbol{\Theta}$ as the product $\boldsymbol{\Phi}\boldsymbol{\Psi}$, then the RIP presented in Eq. (8) will become:

$$(1 - \delta_k)\|\boldsymbol{\alpha}\|_{l_2} \leq \|\boldsymbol{\Theta}_k\boldsymbol{\alpha}\|_{l_2} \leq (1 + \delta_k)\|\boldsymbol{\alpha}\|_{l_2} \quad (10)$$

indicating that orthonormality must be a characteristic of the product between measurement matrix and sparsifying dictionary. Following Eq. (7), if we plan to design a CS CMOS image sensor (CS-CIS), an imager whose output is a sequence of compressed samples usable to solve Eq. (9), then we need to introduce $\boldsymbol{\Phi}$ in its design.

This matrix must be translated into a series of weights that are scalarly multiplied by the output of the pixels in the array. They can be inserted in the sampling sequence using different means i.e. optical elements in front of the array or dedicated circuitry in the CS-CIS itself. Furthermore, these weights will need to comply with the requirements of orthonormality imposed by Eq. (10).

The diagram in Fig. 2 shows the conceptual process of how CS works when applied to image sampling. It also underlines the difference with standard sampling presented in Fig. 1. In this example, each pixel value x_j is multiplied by a coefficient of the measurement matrix $\varphi_{i,j}$. These products are summed together using an analog adder to create the compressed sample and only then this sample is digitised. In this sense each row of the matrix can be seen as a waveform and the resulting sample is a convolution of the image elements by such waveform. It is necessary to swap the coefficients employed, which belong to a row of the measurement matrix, with those of other rows in the matrix to further generate other compressed samples.

Note that, using compressed samples to capture the content of an image eliminates the need to implement an OAAF to avoid spatial aliasing which was a direct consequence of the sampling paradigm based on interpolation Eq.(1) and the Petersen–Middleton theorem and its practical implementation presented in Eq. (3). As such, design choices that determine pitch, number and placement of the pixels and their relationship to image quality derived from this theory do not hold anymore: remember that CS makes a connection between pixel of a sensor and pixel as elements of the image that is not biunivocal.

A priori, if we wanted to sample a specific image, by knowing its structure and formulating opportune assumptions, we could design a dedicated matrix to optimise its sampling. Following

this line of thoughts, there can be an infinite number of possible measurement matrices that could potentially be used in the design of a CS-CIS, each working at its best on a particular type of images, but it is advisable to choose a design strategy that creates a CS-CIS able to function in all situations.

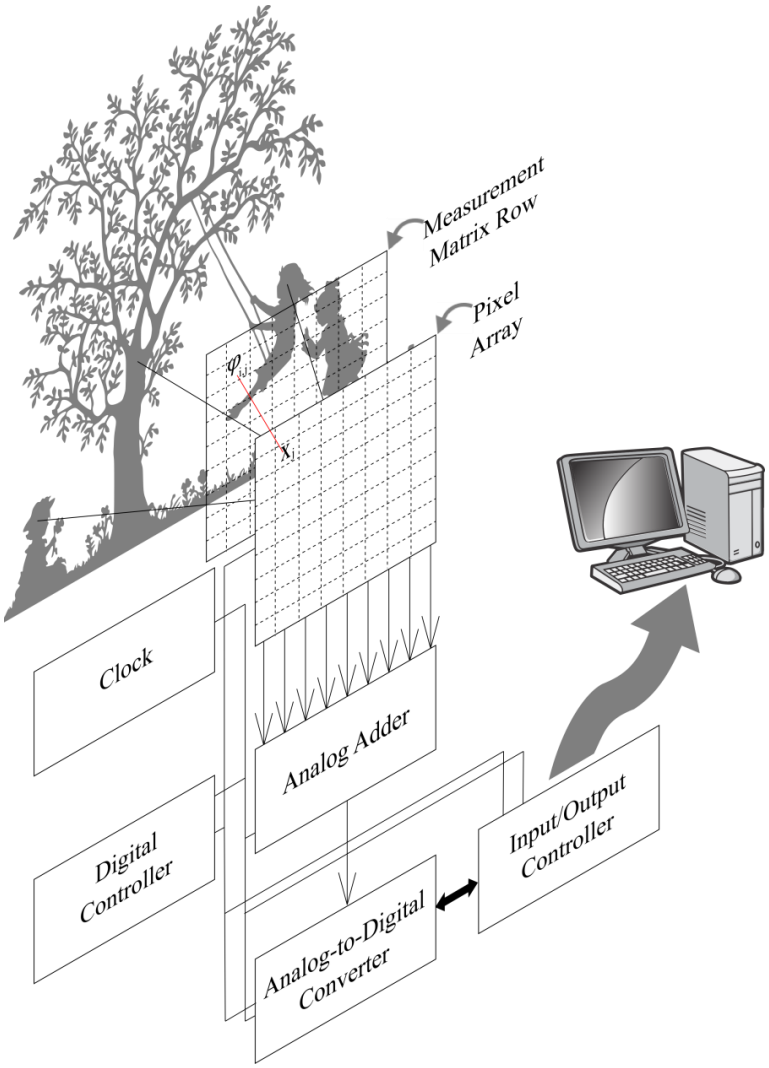

Fig. 2: Compressive Sampling Image sensor diagram.

However, the compressed samples obtained by a CS-CIS are taken from images sampled in the spatial domain, domain in which they are not sparse. They are encased in electrical quantities in the analog domain and these quantities must have adequate dynamic range to describe the content of these samples. The treatment of these quantities has nothing to do with how CS uses **Φ** to reorder the information content of the sampled image: at sensor level, beyond reducing the number of pixels by creating linear combinations, it has no other purpose. Even though sparsifying dictionaries can help a posteriori reconstruction, the quality of the information included in the samples taken by a CS-CIS depends entirely on adequate digitisation.

Digitisation that is carried out using standard components and techniques, which must abide Eq. (2) and all of the considerations derived from the Nyquist–Shannon theorem and which needs to use large digital words to maintain an appropriate resolution after combining the values of several pixels in a large compressed sample.

Among RIP-compliant measurement matrices that can be applied to sample all sorts of signals we find normalised Gaussian matrices. These matrices have entries that are independent standard normal random variables and are some of the most efficient universal measurement matrices that can be employed in CS [Vers09]. Although these matrices would be ideal, in most cases they are not practical: as shown in Fig. 2 each compressed sample is a linear combination of the weighted readings of all pixels (which are N in number) and that means that they are the sum of N products of $\varphi_{i,j}x_j$. The problem of compresses sample quantisation has already been studied in theory[Bouf08][Jacq11] but less attention has been given to the perspective of ADC dimensioning: The upper bound on the amount of bits required to digitise these linear combinations is:

$$\mathrm{B_{CS}} = \lceil \log_2 N \rceil + (\mathrm{B_\Phi} + \mathrm{B_I}) \tag{11}$$

where $\mathrm{B_{CS}}$ is the number of bits needed to digitise a compressed sample, $\lceil \cdot \rceil$ denotes the smallest integer greater than the argument and $\mathrm{B_I}$ and $\mathrm{B_\Phi}$ the number of bits used to quantise the pixel dynamic range and the coefficients respectively. Standard imagers usually employ 8 to 10 bits ADC to digitise the value of a single pixel, if we were to assign the same amount of bits to each coefficient and we had a 64×64 pixel array we would need a converter with at least 28 bits of resolution to properly digitise a single compressed sample. It is virtually impossible to design ADC with such resolutions in standard technologies.

For this reason the use of binary measurement matrices and block-based compressive sampling (BCS) [Gan07], that divides a pixel array into smaller sub-arrays that are sampled digitised independently from one another, is essential for the implementation of measurement matrices in CS-CIS. These binary matrices are obtained using sub-Gaussian distributions: instead of drawing each entry from a normal random distribution they are selected at random to be either 1 or 0. Another important advantage of BCS is that using the same measurement matrix for each block can be exploited to accelerate reconstruction [Akb218] and parallelise measurements [Oike13].

Although this solution is the most common in the design of CS-CIS [Waki06], [Ferg06], [Maji10], [Oike13], [Leit18], [Lee18], there are two alternatives to this choice. The first is to design deterministic binary matrices; i.e. cyclic matrices [DeVo07], sparse bipartite graphs

[Xia15] or other forms of deterministic matrices that try to match the performance of Gaussian random matrices [Jafa12]. And the second is to introduce a measurement matrix using optic elements in which case they use lenses to redirect or diffuse the light either deterministically [Wang10] or at random [Ferg06].

The first step to design a CS-CIS would be to study which solutions have been employed in real prototypes and a good way to categorise these prototypes would be to focus our attention on which elements their designers have used to introduce the measurement matrices as well as if these matrices are deterministic or not.

Using this division we can create two wide categories:

- The coefficients of $\mathbf{\Phi}$ are introduced using optic elements: the measurement matrix is built using different types of lenses so that a standard imager can be used.
- The coefficients of $\mathbf{\Phi}$ are inserted on-chip: the measurement matrix is incorporated as part of the imager design as such it alters the pixel structure and imager architecture.

1.1 Measurement Matrices generated through Optic Elements

The designs of CS-CIS that belong to this category are clearly inspired by the parallelisms that can be found between the sampling theories presented in Eq. (3) and Eq. (5). Early CS-CIS prototypes were designed using optic elements. In a way this is not surprising because, as we have shown in Fig. 2 and Eq. (5), the implementation of a measurement matrix in practice is none other than the substitution of the conventional OLPF needed to avoid spatial aliasing by a matrix of random coefficients. The Single Pixel Camera [Waki06] represents one of the first two prototypes of CS-CIS. It introduces the waveforms or coefficients that create the compressive samples through the use of an array of steerable micro-mirrors controlled through a pseudo random number generator (PRNG) Fig. 3.

This prototype implements a measurement matrix efficiently through the use of optical elements. In this design a lens focuses the scene on an array of movable micro-mirrors. Each micro-mirror can be oriented to reflect a portion of the scene towards the single photodiode of the camera. The sum of the light of all the parts of the scene that converge on the photodiode form a compressed sample in the optical domain that is then transformed into an electrical signal by the pixel. To reconstruct the sampled image, the micro-mirrors take on the role of a sub-Gaussian distribution of binary coefficients.

This example is the embodiment of both the advantages and disadvantages of CS-CIS design with respect to conventional sampling. On the one hand it is apparent that, since there is no spatial aliasing to consider, CS does not impose restrictions on the positioning of the active

elements of the sensor allowing more freedom in their design, so much so that it is possible to reduce their number up to a single unit. But, on the other hand it makes it clear that the information incorporated in a single analog signal is that of many pixels, where in this instance we intend by pixels the values of the vector of unknowns **x**, as such, the ADC that follows will need a higher resolution than that required to digitise a pixel of a standard imager, because remember, as we have seen in Eq. (11), CS only compresses space-wise and this does not affect the amount of bits needed to digitise the electric signals produced by the sensing elements.

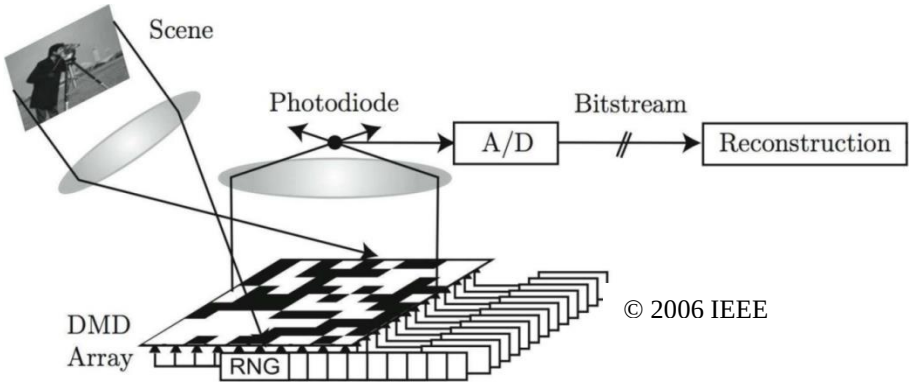

Fig. 3: Single Pixel Camera, Image taken from "An Architecture for Compressive Imaging" [Waki06] © 2006 IEEE.

Another important consideration to make is that the compressed samples are taken one after the other forming a temporal sequence even though they will be used to recover the same still image. After each sample, the array of micro-mirrors has to reconfigure itself into a new position in order to be able to grab the next sample which must be different from the previous. This consideration that seems almost trivial creates a strong limitation on the design of CS-CIS: the process of taking compressed samples lengthens the times in which the image must be kept unchanged. It is true that in standard imagers we need to sweep the entire array, a process that is usually carried out in row-wise fashion, and that during that time the image has to remain stationary but once that is done the image is captured. In CS one needs to sweep the entire array (or in this case reconfigure the micro-mirrors) once for each sample and this operation potentially makes the acquisition process much longer than in standard imagers. Either we become able to ideally collect all samples simultaneously or the CS-CIS should operate at a speed that is several orders of magnitude faster than a standard imager would with respect to the fastest movement contained in the scene to be captured.

Another example of optical measurement matrix, which was reported almost at the same time as the Single Pixel Camera, is the Random Lens Camera [Ferg06] invented at the Massachusetts Institute of Technology (Fig. 4). This prototype, instead of pseudo-randomly generating each

waveform sequentially, as the Single Pixel Camera does, it uses a shattered lens to refract and diffract the incoming light randomly and thus creating all the measurement waveforms at once.

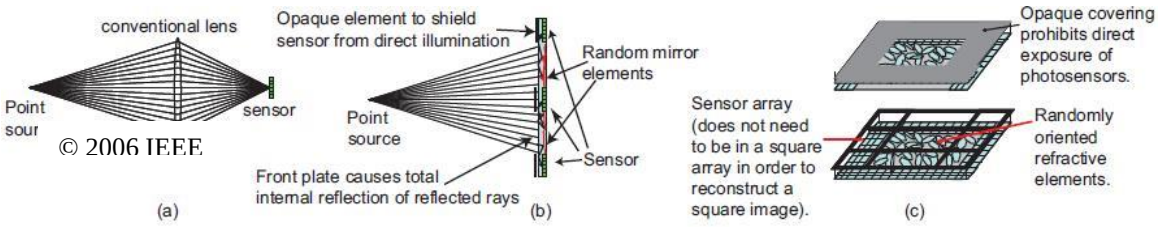

Fig. 4: Random Lens Camera, Image taken from "Random Lens Imaging" [Ferg06] © 2006 IEEE.

This prototype is the one that more closely approaches the RIP requirement as presented in Eq. (8) because it does not use an approximation such as PRNG to cycle through deterministic rows of a binary measurement matrix. It uses a truly random shattered lens that, when viewed from different points on the focal plane, creates random waveforms where the active elements of the sensor are placed.

Furthermore, since various pixels are used to collect the compressed sample simultaneously, it resolves the burden on the operating speed given by the requirement of time invariance of the sampled scene. But, whereas this solution resolves these problems, it introduces new one, calibration: in a randomly broken lens there is no telling how the light will be scattered. This implies that there is no way of knowing, a priori, the coefficients of the measurement matrix that this lens creates, coefficients that, in the end, will still be needed for reconstruction. Hence, before using this camera we must recover them. To do so we must present a number of known images equal to the number of compressed samples generated by the camera itself and use its outputs to invert Eq.(5) and calculate all of the unknown coefficients.

This process, besides the obvious difficulty posed by precisely presenting the known images to the sensor can also be very tedious because, in order to calibrate a 128×128 pixel array, we would need to use 2^{14} images. What aggravates the situation even more is that these coefficients will be numbers other than ones and zeros thus posing a heavy burden on the ADC (see Eq.(11)).

This prototype is another fine example of how the measurement matrix can be successfully implemented using optic elements but it also helps us to stress the importance of using approximations such as the PRNG introduced by the Single Pixel Camera. While it is true that using less than optimal matrices negatively impacts the overall performance of the CS-CIS in terms of RIP, in reality it is the choice to make in order to ease ADC requirements and to allow us to generate identical matrices both during acquisition and at reconstruction without needs for calibration or constant transmission of the matrix coefficients.

Besides these two pioneering examples of measurement matrix implementation through

lenses and mirrors, there have been many more prototypes in these past years that have exploited optical elements to reach this goal. One of them, for instance, is the Super Resolution Imager [Wang10] that uses as matrix a fish eye distortion lens, which can be described as a circulant matrix, and exploits its orthonormality [Wota10] in order to generate RIP compliant compressed samples. Another uses the properties of scattering materials [Liut14] in a similar way to how [Ferg06] used a shattered lens. Following the example of [Waki06], [Chen05] implements a similar solution for imagers meant to sample in the mid-infrared spectra. Many others [Roum08], [Arce14] or [Moch15] make use of coded apertures to reproduce the ones and zeroes of binary matrices and some even mount an imager on top of a platform connected to two micro-motors that pseudo randomly move up and down and sideways to create random exposure [Shi09].

All of these CS-CIS have been successful in implementing diverse measurement matrices in order to achieve compressed sample extraction and they show how much freedom the lack of concern for aliasing effects can bring to an imager design. But they all suffer from incredible burden during A/D conversion and some of these matrices cannot be easily reproduced both at sensor level and during reconstruction. The curvature of a fish eye lens such as the one used in [Wang10] might be difficult to translate into coefficients and the delays of the elements used to code the aperture of the imager or to set it in motion might induce errors that are difficult to predict and to correct. For this reason, the majority of the efforts have focused on inserting the measurement matrix coefficients directly on-chip incorporating it as part of the imager design.

1.2 ON-CHIP MEASUREMENT MATRICES

As we have illustrated with some examples in the previous section, two major limitations arise while incorporating CS measurement matrices in a sensor design. These limitations also apply when, instead of using optic elements, these matrices are implemented using electric components at sensor level. First of all, the coefficients of such matrices must be known at both extremes of the communication channel. This is, not only the sensor must know which compressive strategy we use in order to generate the corresponding compressed samples, but the same strategy must be known also at the end of the channel, in order to reconstruct the original image. Therefore, either the matrix that we choose is generated at the sensor and transmitted to the image reconstruction system or it needs to be stored at both ends. The second question is dynamic range. As we have described in Eq. (11), if we collect compressed samples using binary measurement matrices, the number of bits required to describe these linear combination of N pixels would be $\log_2 N$ in addition to the bits chosen to describe one single pixel. As we will discuss in Chapter 3, this is a major limitation for an implementation of this addition in the

analog plane, as dynamic range in analog circuits is limited by the available output range and noise level.

To try to comply with the first requirement, in signal processing literature, there are many works that study deterministic binary measurement matrices; i.e. cyclic matrices [DeVo07], sparse bipartite graphs [Xia15] or other forms of deterministic matrices that try to match the performance of Gaussian random matrices [Jafa12] and that can be easily known by both the sensor and the reconstruction system. These matrices are usually coined hardware-friendly by their authors because of to their binary nature but they still partially disregards the fact that, on-chip, they need hardware resources to be generated, transmitted and/or stored. To store a binary measurement matrix in its integrity a large memory would be required, which is not practical. If, on the contrary, we were to introduce a transmission between sensor and reconstruction system to continuously send the rows of the matrix, the amount of exchanged data would void CS benefits as a compression technique.

As CS is an acquisition technique, the fact that we cannot apply these theories to actively design sensors is, in itself, the limitation of these works. For this reason, measurement matrices designed in analog microelectronics [Maji10], [Oike13] mainly focus on the use of PRNG to recursively create the rows of the matrix sample after sample. This solution employs the minimum possible amount of on-chip resources and does not need data feedback between sensor and reconstruction system.

One of the early examples of CS-CIS prototypes with on-chip generated measurement matrix is the CMOS Imager/Compressor designed at the École Polytechnique Fédérale de Lausanne [Maji10]. As it was the case with the Single Pixel Camera [Waki06] and the Random Lens Camera [Ferg06] presented in the previous section, this example embodies both the advantages and disadvantages of CS-CIS design with respect to conventional sampling. The matrix is generated by introducing 1-bit registers in each pixel and connecting them in sequence to create three distinct Linear Feedback Shift Registers (LFSR) that form a programmable two-dimensional scrambling technique that guarantees the randomness of the matrix binary coefficients Fig. 5 [Maji10].

This example shows the two disadvantages that, in general, CS implementation on-chip has with respect to optical implementations or to standard imaging. First of all we notice that, unless we encode the initial seed in the registers reset [Guic16], when the matrix is introduced on-chip, there is the need of an initialization phase in which an initial set of coefficients, known as seed, is sent to all of the registers. If a mistake were to occur during this phase the sampled image would be irretrievable. The second is that, introducing extra components within the pixel array

reduces the pixels fill factor, that is, the ratio of a pixel light sensitive area to its total area, thus worsening the quality of the sampled image.

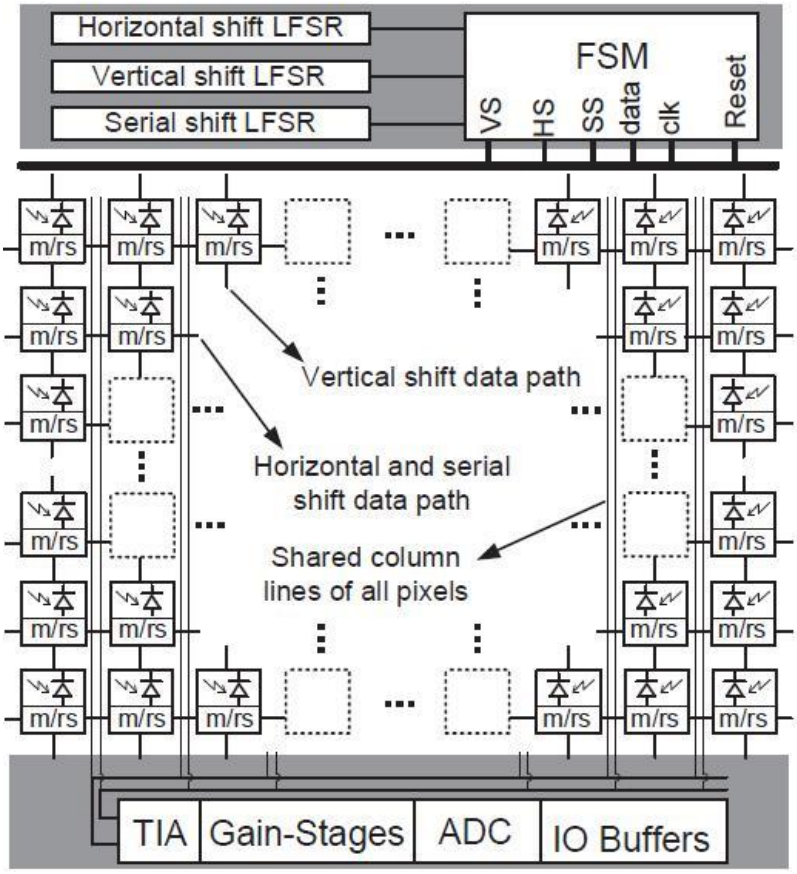

Fig. 5: Architecture of "A (256x256) Pixel 76.7mW CMOS Imager/Compressor Based on Real-Time In-Pixel Compressive Sensing", Image taken from [Maji10] © 2010 IEEE.

On the other hand, this architecture also shows the advantages that, in general, incorporating a measurement matrix on-chip has. A matrix generated this way makes it so that each coefficient is directly tied to its corresponding pixel; there is no need to translate the movements of mechanical components with unpredictable delays into coefficients and there is no need to fine tune how the light that passes through a deforming lens reaches each element of the array. This type of implementation also offers a certain degree of flexibility in that by switching the initial seed it is possible to easily generate a completely different measurement matrix.

But, the most important consideration to make is that each pixel only has to capture the light that belongs to a specific part of the scene. When the measurement matrix is created using optic elements, what each pixel of the array collects, in terms of incoming information, is already the linear combination of various parts of the scene that is being sampled. In this new scenario, on the other hand, after the light has been converted into analog signals by the pixels, the compressed samples have yet to be formed. This fact can be used to alleviate the second problem

that all CS-CIS face: the dynamic range needed to describe the signals. In this example for instance the authors split the contributions of the pixels, operating in current mode, into two separate current summation lines which are then used as differential inputs of a transimpedance amplifier (TIA). This differential TIA provides a differential output voltage. The corresponding compressed samples that this generates are made by sums of pixels multiplied by binary coefficients that, instead of being ones and zeros, become ones and minus ones and thus, thanks to the differential TIA that rests the contributions of a line to those of the other, could be described with fewer bits.

The problem of this solution is that, since the measurement matrix is pseudo-random, the amount of coefficients that are ones or minus ones constantly fluctuates; moreover it is possible that the light incident on the pixels of one of the two sets could be much brighter than that of the other. These two uncertainties combined make it so that the ADC that follows the TIA should be designed taking into consideration the worst case scenario, i.e. the coefficients being either all positive or negative with a difference in illumination between the two possibilities as wide as possible. In this particular case this would leave us with a 25-bits ADC: following Eq. (11), 16 bits would be needed for the N amount of pixels in the array (which is 256×256), 8 bits are needed to describe the content of a single pixel and 1 bit is needed for the sign.

A more pragmatic and widely extended approach is that of BCS [Gan07]. Architectures that use this design technique [Oike13], [Kali14], [Dadk15] have their arrays divided into macro-blocks onto which the measurement matrix is applied separately and whose outputs are digitised independently block by block. An example is presented in Fig. 6 [Oike13].

In this example, a ΔΣ Single-Shot CS-CIS is designed to perform CS using a row block/pixel selector and column block selector multiplexers both controlled by PRNG and a series of column-parallel ΔΣ ADC with decimation filters and a column output scanner to generate more compressed samples simultaneously. This prototype clearly shows how the application of BCS can reduce the amount bits needed to properly convert the samples from the analog domain to the digital one. This statement is easily understood when we consider that a digitisation carried out block by block reduces N, the amount of pixels portrayed in Eq. (11).

Another advantage that is present in this prototype with respect to the previous one is that, whereas early implementations of CS-CIS, such as the one presented in Fig. 5, were carried out by embedding a register inside each pixel, this solution replaces the registers by simple and more compact selectors. These selectors are then driven by LFSR that surround the array. This simplification is possible because, at pixel level, we can consider random row and column selection as the multiplication of two independent binary random variables. The multiplication of

random variables still delivers random outputs.

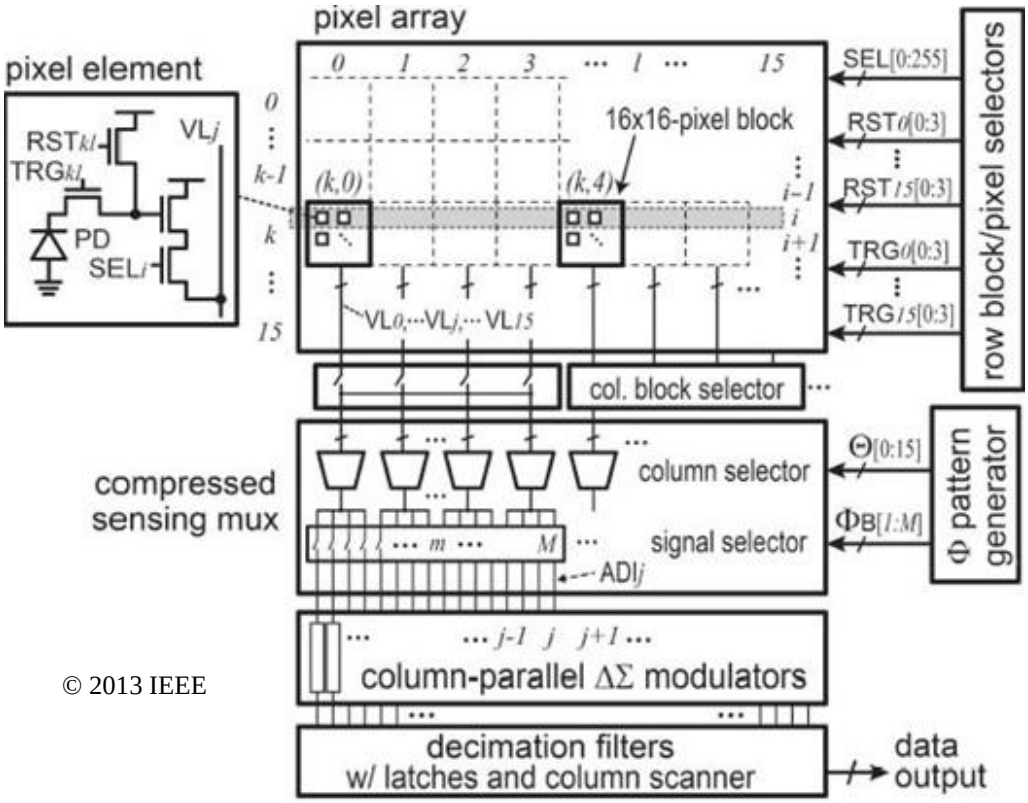

Fig. 6: Architecture of "CMOS Image Sensor With Per-Column ΣΔ ADC and Programmable Compressed Sensing", Image taken from [Oike13] © 2013 IEEE.

In general, selectors can be implemented with few transistors, using for instance switches or gated inverters and as such, their use, besides reducing the amount of on-chip resources needed to generate a measurement matrix it also increases the fill factor of the pixels. This last benefit though, in most cases, might only be apparent because, even if the fill factor is increased, the division of the array in small blocks introduces asymmetries that still worsen the overall image quality.

The use of ΔΣ converters has become fairly popular in CS-CIS design to improve reconstruction accuracy [Günt10] and it shows promising results [Lee18] but it is not the only method used to generate compressed samples. As [Leit18] reports there are various possible architectures that can be used to implement BCS. These architectures are usually defined by the choice of which electrical magnitude is employed to translate light intensity into analog signals. Pixels adopted in CS might have their outputs summated as currents [Jacq09], [Maji10], as charge [Kati13] or as voltages [Oike13], [Leit18], [Lee18].

Despite the various possibilities available to the designer, it is important to remember that the overall the size of the blocks is at the essence of the quality of the sampled image. We have shown [Trev20] that fast and accurate reconstruction algorithms opt for small blocks, typically

8×8 but, on the other hands, blocks that are too small deteriorate the quality of the sensor introducing artefacts within the array and during reconstruction. We have established that, blocks which are 32×32 in size are, de facto, the best compromise between the two divergent necessities. Considering these results, even if BCS somehow helps to reduce the dynamic range of analog signals it does not reduce it enough so that 32×32 blocks can be easily handled conventionally. Furthermore, it is important to notice that LFSR as other forms of PRNG produce binary matrices that are neither normalised nor orthogonalised such as those that the RIP requires.

1.3 THESIS ORGANIZATION

Studying the prototypes presented in the previous section we can detect a clear pattern. Most of these works have been developed by research groups affiliated to universities or companies with a strong focus on circuit design. This is to be expected, and as a consequence of this tendency most of the published reports focus on the CS-CIS performances from an electrical viewpoint, meaning, these circuits have been characterised in terms of power consumption, speed of operation, number of transistors per pixel, fill factor, frame rate and such. This creates a lack of information with respect to the theoretical performance of matrices generated by PRNG in terms of RIP.

This is largely due to the fact that this sort of matrices, in the field of signal processing, arises little to no interest due to their simple nature. We will dedicate the second chapter of this dissertation to formalise a method to study the performance of these generators from the perspective of RIP introducing some mathematical tools to characterise the performance of different kinds of PRNG not only from an electrical viewpoint but also from a mathematical one.

Furthermore, thanks to the joint work carried out at the Institut Supérieur d'Électronique de Paris in cooperation with a research group mainly focused on CS reconstruction algorithms and reconstructed image quality [Trev20] we have defined a CS-CIS architecture that allows us to increase the precision of compressed samples digitisation by means of pulse modulation. This approach substitutes the more commonly employed analog to digital conversion in favour of pulse width modulation thus transforming the lack of dynamic range usually found while digitising large blocks into a problem of time consumption. This solution might decrease the overall frame rate of a circuit but in exchange it increases the precision of the compressed samples that it delivers as well as the size of the blocks of a CS-CIS build upon BCS theory thus also increasing the reconstructed image quality.

The fourth chapter of this thesis will present the materialisation of the findings from chapters

two and three. In this part we will introduce a CS-CIS imager which is 64×64 in size and that uses an Elementary Cellular Automaton (ECA) for the generation of the measurement matrix and pixels pulse width modulation to create compressed samples in the digital domain as the sum of a sequence of pulses. This chapter will also include a detailed report on the laboratory setup used to study the performance of our imager i.e. chip encapsulation and choice of Field-Programmable Gate Array (FPGA) and on the tests performed to evaluate its correct functioning and performance.

By taking into account the limitations presented in chapters two and three, the last three chapters of this thesis are dedicated to the study of hardware-friendly applications of CS that could be readily applied to CS-CIS prototypes already present in the state of the art.

Chapter five introduces a non-recursive algorithm that exploits a differential readout method that transforms binary measurement matrices into matrices of ones and minus ones, such as the one presented in [Maji10], to pre-process the information contained in a set of compressed samples non-recursively in order to understand, in a context of video surveillance, if the presence of a movable object within a frame warrants its reconstruction or the extraction of its features.

Trying to go beyond the limitations of binary matrices we will propose in chapter six a possible way of using PRNG and differential readout methods to create ternary matrices with better performances than their binary counterparts.

We will dedicate chapter seven to present a framework meant to transform traditional feature extraction algorithms based on linear transformations into ready to use sparsifying dictionaries to aid reconstruction algorithm by lowering the amount of relevant elements of the signal that they need to recover. We hope that, by using these feature-extracting sparsifying dictionaries, it will be possible to use reconstruction algorithms to extract features by transforming these very features into the non-zero elements that the algorithm has to recover. We will use as example the Harris corner detection algorithm.

We will summarise our conclusion in the eighth and last chapter.

CHAPTER 2
PRNG-GENERATED MEASUREMENT MATRICES

PRNG are non-linear spatially discrete and temporally discrete dynamic systems made of binary logic elements that show high sensitivity to initial conditions and evolve in time according to a divergent and fractal behaviour [Boei16]. A new state in their time evolution is derived from their actual state using a feedback mechanism that promotes this instability. In the prototypes of CS-CIS studied in section 1.2, each temporal state is used to generate a row of the binary measurement matrix recursively. It is important to remember that, since each row of a measurement matrix is generated from the previous row, there is no need to store the whole binary matrix on-chip or to receive it from an outside source.

PRNG commonly used to implement measurement matrices into CS-CIS is the Linear Feedback Shift Register (LFSR) [Sara12], [Mazz08]. LFSR consist of a series of cascaded flip-flops (Fig. 7). Some of the outputs of these flip-flops, besides being connected to the input of the flip-flop that they precede, are also connected, by means of XOR logic gates, to the input of the very first one. When considered individually, the output Q of each flip-flop in the sequence evolves in time with a behaviour that resembles a Bernoulli probability distribution with probability $P = 0.5$.

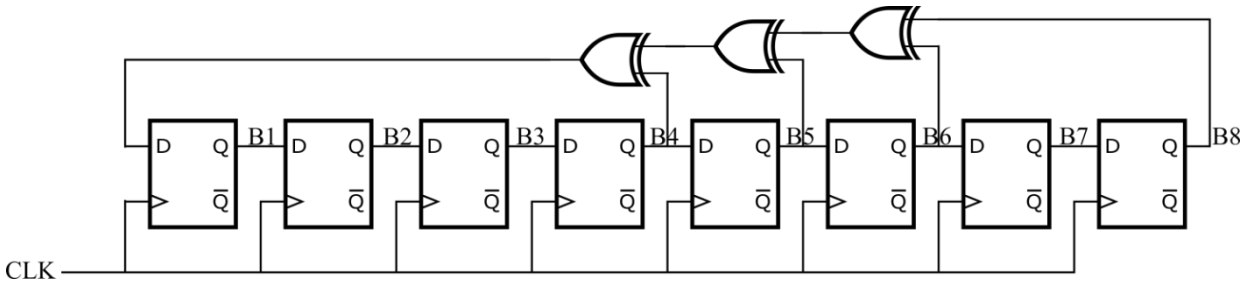

Fig. 7: Example of an 8-bit LFSR.

Even if LFSR are good candidates to emulate sub-Gaussian random probability distributions thanks to their low impact on area consumption, ultimately the matrices that they produce row-

by-row recursively still follow a deterministic pattern. For this reason it is necessary to analyse this pattern to understand how well the generated measurement matrix approximates the functionality of one that is extracted directly from a real binary probability distribution. Moreover, LFSR are not the only example of circuits that can serve as PRNG in CS-CIS. A valid alternative are Elementary Cellular Automata (ECA) [Jen90].

By definition a Cellular Automaton (CA) is a spatially and temporally discrete dynamical system made of identic individual cells that evolve in time according to a common rule [Wolf83]. A new state in the discrete time evolution of a cell is derived from this rule and it only depends on local spatial conditions. These local conditions are the actual state of the cell itself and the states of its neighbouring cells. To generate a pseudo-random binary pattern to be used as measurement matrix, we will focus on the simplest possible implementation of CA that takes the name ECA (Fig. 8).

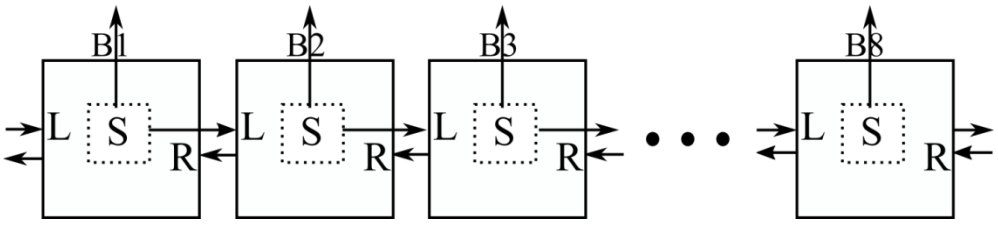

Fig. 8: Example of an 8-bit ECA.

We will study possible hardware implementations for these PRNG as well as the matrices that they generate in order to establish a basis for comparison among them both in terms of quality of the generated matrix and of hardware resources needed for their implementation.

2.1 MEASUREMENT MATRIX GENERATION THROUGH ECA

Since the cells of an ECA are connected sequentially in a one dimensional fashion (Fig. 8), their evolution in time depends only upon three parameters: their own current state (S) and those of their two closest neighbours (L, R). For this reason, considering that each cell of an ECA is a three-input one-output binary element plus a flip-flop, there are 256 possible combinations for these three parameters [Wolf83]. These configurations are defined as rules and the number assigned to each rule corresponds to the decimal representation of the binary sequence of outputs that the system can produce given all possible input combinations, e.g. Fig. 9 represents all the combinations for a cell that evolves according to rule 30.

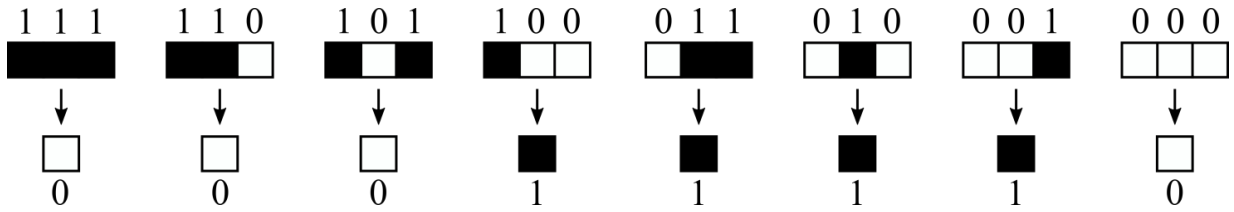

Fig. 9: Evolution pattern of a cell of an elementary cellular automaton implementing rule 30.

It is immediate to see that the sequence '00011110' is the binary representation of the decimal number 30. When Steven Wolfram first introduced the concept of ECA [Wolf83], he divided these rules into four classes depending on the type of geometrical pattern that their temporal evolutions produced:

- Class I: homogeneous geometrical patterns;
- Class II: periodic geometrical patterns;
- Class III: random-like chaotic geometrical patterns;
- Class IV: local complex geometrical structures that move in time and space.

We will focus our attention on rules that belong to Class III and we will study their dynamic behaviour along with the behaviour of LFSR to establish which PNGR is better suited to approximate a sub-Gaussian probability distribution. But first, to set our goal clearly we need to define the quality of the compressed samples that an ideal PRNG can potentially produce and to do so it is necessary to analyse these binary matrices in terms of RIP.

2.2 RIP OF PRNG GENERATED MEASUREMENT MATRICES

Suppose that an area of size $\sqrt{N} \times \sqrt{N}$[1] pixels is sampled by a measurement matrix generated row-by-row as is the case of the prototypes presented in section 1.2. Let us consider the simplest possible case in which an element of the PRNG is assigned to generate recursively the coefficients of one pixel of the imager so that each row of the measurement matrix $\mathbf{\Phi}$ corresponds to a time step in the discrete evolution of the generator itself. Mathematically, this sampling process is formulated as $\mathbf{y} = \mathbf{\Phi x} + \mathbf{n}$, where $\mathbf{x} \in \mathbb{R}^N$ is the $\sqrt{N} \times \sqrt{N}$ array of pixels folded in vector form and $\mathbf{n}$ is an additive white Gaussian noise which standard deviation is estimated to be ε (Eq.(5)) at the reconstruction stage. The reconstruction of the original signal $\mathbf{x}$ from the compressed samples $\mathbf{y}$ is an ill-posed problem. By making use of sparsity, a well-known characteristic existing in natural signals, a *k-sparse*[2] signal, like images and videos, can be reconstructed from a few samples obtained using an appropriate measurement matrix.

A sufficient condition for the unique and exact recovery of the signal is the RIP of the

[1] A requirement of CS is that the image be square. The N pixels, where $\sqrt{N} \in \mathbb{N}$, for purposes of reconstruction are treated as vector $\mathbf{x}$

[2] $\mathbf{x}$ is called *k-sparse* or a signal with sparsity level k, if it has no more than k non-zero components with $k \ll N$.

measurement matrix $\mathbf{\Phi}$ [Cand06] that for the sake of clarity I will report once again to expand on the concept. A matrix $\mathbf{\Phi} \in \mathbb{R}^{M\times N}$ satisfies the RIP of order k if there is constant $\delta_k (0 < \delta_k < 1)$ such that, for all vectors $\mathbf{x} \in \mathbb{R}^N$ with $\|\mathbf{x}\|_0 \leq k$ (i.e. *k-sparse* signals[1]), it holds:

$$(1-\delta_k)\|\mathbf{x}\|_2^2 \leq \|\mathbf{\Phi x}\|_2^2 \leq (1+\delta_k)\|\mathbf{x}\|_2^2 \tag{12}$$

The smallest non-negative value for δ_k is called restricted isometry constant (RIC) of order k. The construction of a measurement matrix which satisfies the RIP is a central problem in CS. If sparse signals with maximum possible sparsity level k can be recovered exactly and stably, it is said the measurement matrix $\mathbf{\Phi}$ has sparsity order k. An upper bound for the sparsity level is $k \leq CM/\log(N/M)$ where C is a constant [Indy08]. The signals with k above this bound can only be approximately reconstructed. In practice, to recover a signal $\mathbf{x}$ with a large k, is desirable to have a measurement matrix with a δ_k as small as possible. The limit imposed on the RIP of random binary measurement matrices has already been studied in [Lu13] Such binary matrices satisfy the RIP with:

$$\delta_k = \frac{(3k-2)m}{(k-2)m+4d} \qquad \text{if } 3 \leq d \leq M/2 \text{ and } 2 \leq m \leq d \tag{13}$$

where d is the estimated number of non-zero elements in each column of $\mathbf{\Phi}$ and m is the maximum inner product between two distinct columns. The distribution of 1's among the elements of random binary matrices, generated by pseudo random generators, can be associated with a Bernoulli probability distribution in which each element has a probability P of being 1 and $\overline{\mathrm{P}} = 1 - \mathrm{P}$ of being 0. Since these pseudo random measurement matrices try to emulate the normalized Gaussian distribution, which is a symmetric probability distribution, usually, circuit designers choose LFSR because their distribution is akin to $\mathrm{P} = \overline{\mathrm{P}} = 0.5$.

Let us consider the ideal case of an ideal PRNG capable of recursively producing independent binary elements with a symmetric Bernoulli probability distribution having $\mathrm{P} = 0.5$. If we consider a large enough pseudo random binary generator (a large enough number of pixels N), since each of its elements, by construction, is independent from the others and the probability distribution is symmetrical, we will obtain a measurement matrix $\mathbf{\Phi}$ with an equal amount of 0s and of 1s row-wise. If the acquisition process is long enough (a large enough number of acquired compressed samples M), we will have an equal amount of 0s and of 1s column-wise as well. Such measurement matrix, whose entries are randomly drawn from pseudo random generators, satisfies the RIP with Eq. (12). Following [Lu13], let us define the mutual coherence of $\mathbf{\Phi}$ as:

[1] $\|\mathbf{x}\|_0$ norm denotes the number of non-zero components in **x.**

$$\mu(\mathbf{\Phi}) = m/d \tag{14}$$

We can derive m and d as functions of the number of pixels in the image N, the sampling subrate of the sensor $S = M/N$, and the probability of each element of being different from 0, P. From the definition of subrate (S), it is straight forward that $M = S \times N$. Moreover, using the probability that each element of the matrix has of being different from 0, we can define the number of non-zero elements in a column of $\mathbf{\Phi}$ as:

$$d = S \cdot N \cdot \mathrm{P} \tag{15}$$

Mutual coherence $\mu(\mathbf{\Phi})$ of a matrix represents the maximum absolute value of the cross-correlations among its normalized columns and is defined as:

$$\mu(\mathbf{\Phi}) = \max_{j \neq k} \left| \boldsymbol{\varphi}_j^H \boldsymbol{\varphi}_k \right| \tag{16}$$

where $\boldsymbol{\varphi}_k$ is the k-th normalized column of matrix $\mathbf{\Phi}$ and $\boldsymbol{\varphi}_j^H$ is the conjugate transpose of the j-th normalized column of the same matrix. For a matrix to be column-wise normalized it means that for each column: $\boldsymbol{\varphi}_j^H \boldsymbol{\varphi}_j = 1$. Given the definition of mutual coherence and the randomness of $\boldsymbol{\varphi}_j^H$ and $\boldsymbol{\varphi}_k$, to approximate *m*, we will make use of the most probable outcome of a dot product between normalized columns of $\mathbf{\Phi}$:

$$m = \mathrm{E}\left[\boldsymbol{\varphi}_j^H \boldsymbol{\varphi}_k\right] d \tag{17}$$

Remember that $\boldsymbol{\varphi}_j^H \boldsymbol{\varphi}_k$ is the sum of M element by element products. Each one of these M products will be different from zero if and only if the two elements that are being multiplied differ from 0 as well. By construction, we are multiplying independent random variables; the probability that their product be different from 0 is equal to the joint probability of the single elements. For this reason, the expected amount of non-zeros among the M products is:

$$E\left[\mathrm{Tr}\left(\boldsymbol{\varphi}_j^H \cdot \boldsymbol{\varphi}_k\right)\right] = \mathrm{P}^2 \cdot S \cdot N \tag{18}$$

where $\mathrm{Tr}(\cdot)$ is the trace of the matrix obtained by the vector multiplication of $\boldsymbol{\varphi}_j^H$ and $\boldsymbol{\varphi}_k$. Furthermore we can derive the value of each product in terms of P, N and S by using the definition of normalization and remembering that $\mathbf{\Phi}$ is binary. Normalizing a vector involves dividing each one of its non-zero elements by the Euclidean norm of the vector itself. Since each non-zero element in a binary vector has to take the value of 1, all elements of a normalized binary vector will be equal to the inverse of the vector Euclidean norm itself. For this reason, if

the i-th elements in column $\boldsymbol{\varphi}_j$ and column $\boldsymbol{\varphi}_k$ are non-zeros, then their normalized product will be:

$$\boldsymbol{\varphi}_{j\,i}^{H}\boldsymbol{\varphi}_{k\,i} = \frac{1}{\mathrm{P}\cdot N\cdot S} \tag{19}$$

Combining Eq. (18) and Eq. (19), we can deduce the most probable cross-correlation between two columns of a binary matrix $\boldsymbol{\Phi}$**.** This correlation only depends on P and it is independent from the number of pixels and the sampling subrate:

$$\mathrm{E}\left[\boldsymbol{\varphi}_j^H\boldsymbol{\varphi}_k\right] = \frac{\mathrm{P}^2\cdot S\cdot N}{\mathrm{P}\cdot N\cdot S} = \mathrm{P} \tag{20}$$

Joining Eq. (13), Eq. (14) and Eq. (20) we obtain:

$$\delta_k = \frac{(3k-2)}{(k-2)+4/\mathrm{P}} \qquad \text{if } 3/(S\cdot N) \le \mathrm{P} \le 1/2 \text{ and } 2/(S\cdot N) \le \mathrm{P}^2 \le \mathrm{P} \tag{21}$$

For $\mathrm{P} = 0.5$ and $N \geq 6$, it is possible to see that all of the conditions hold and that random binary matrices that follow a symmetric probability distribution will have RIP ($\delta_k \leq 1$) if and only if $k \leq 4$.

In other words, a recovery algorithm can deliver error free reconstructions only if the sampled image can be described with four or less elements. Eq. (21) poses a harsh limitation on matrices typically used in CS-CIS implementations. For example, an image that could be described with only four non-zero variables could include a monochromatic regular polygon on a monochromatic background. We would need two coefficients to localise its centre (Cartesian coordinates or polar coordinates), another to establish the number of sides (a simple number or the angle that each pair of adjoined sides share) and the last one to determine its size (side length or apothem). Anything more than that would incur in reconstruction errors.

Given this less than optimal result, the least that we can do is to define a set of mathematical tools that can be used to establish which PRNG approximate a binary random distribution as closely as possible. If we failed to do that we would experience an increase in the mutual coherence of $\boldsymbol{\Phi}_G$ Eq. (14) breaking the upper limit of the first condition imposed on Eq. (13) that would not allow $\boldsymbol{\Phi}_G$ to respect RIP in the first place.

To avoid adding on top of Eq. (21) more errors due to non-idealities in our design choices, let us define $\boldsymbol{\Phi_G} \in \{0,1\}^{M\times N}$ as a binary measurement matrix obtained using a PRNG and $\boldsymbol{\Phi_R} \in \{0,1\}^{M\times N}$ as a binary measurement matrix obtained selecting each element at random from a binary probability distribution with $\mathrm{P} = \overline{\mathrm{P}} = 0.5$. $\boldsymbol{\Phi}_G$ will be a good approximation of $\boldsymbol{\Phi}_R$ if it

holds three characteristics:

- the number of non-zero elements in a row of $\mathbf{\Phi}_G$ must approximate P ≈ 0.5;
- the temporal evolution of the elements of the PRNG do not present repeating patterns;
- the temporal evolution of the elements of the PRNG present no correlation with one another.

The first and second requirements bind the number of non-zero elements in $\mathbf{\Phi}_G$ with the average of the probability distribution used to create $\mathbf{\Phi}_R$. The second and third requirements can be used to bind the mutual coherence of $\mathbf{\Phi}_G$ as:

$$E[\mu(\mathbf{\Phi}_G)] \approx E[\mu(\mathbf{\Phi}_R)] = \mathrm{P} \tag{22}$$

We will analyse how binary measurement matrices generated using LFSR and Class III ECA fulfil these criteria in order to compare their performances so to make an educated choice on the type of PRNG that better fits into the design of a CS-CIS. Furthermore, since there are 26 different rules of Class III ECA, we will use these same mathematical notions to discriminate among their performances in order to understand if all Class III rules can be used to generate CS measurement matrices or if some rules behave better than others.

2.3 POWER SPECTRAL DENSITY

To understand if the rows in $\mathbf{\Phi}_G$ present repeating patterns and to compare the dynamical behaviour of these PRNG we will use Power Spectral Density (PSD) analysis. This technique has been extensively applied to analyse discrete dynamical systems such as ECA [Nina08] or LFSR [Mazz08]. For a spatially discrete and temporally discrete dynamic system, the Discrete Fourier Transform (DFT) can be expressed as:

$$\phi_h(f) = \frac{1}{M}\sum_{i=1}^{N} \varphi_h(t) e^{-j2\pi t f / M} \text{ where } t = 1,2,\cdots,M \tag{23}$$

where, following a notation similar to [Nina08], $\phi_h(f)$ is the DFT value of the h-th element of the PRNG at frequency f, M once again is the number of discrete time steps of the PRNG as well as the number of rows in $\mathbf{\Phi}_G$ and $\varphi_h(t)$ is the state of the h-th element of the PRNG at time t as well as the t-th element of the h-th column of $\mathbf{\Phi}_G$.

PSD expresses the distribution of the energy of a waveform among its different frequency components. Any peak in a graphic of PSD(f) over f at a given frequency f_p would represent a strong repeating pattern of period $1/f_p$ among the rows of $\mathbf{\Phi}_G$ Given all $\phi_h(f)$ PSD can be computed as:

$$\mathrm{PSD}(f) = \frac{1}{N}\sum_{i=1}^{N}|\phi_h(f)|^2 \tag{24}$$

where N is the number of elements of the PRNG as well as the number of columns in $\boldsymbol{\Phi}_G$. The PSD profile of a suitable and efficient PRNG for CS should closely resemble white noise since its energy should equally spread throughout the entire spectrum.

2.3.1 Power Spectral Density of ECA

We devised a MATLAB experiment in which we evolved 64-cells ECA of the 26 possible Class III rules over a period of time, Fig. 10 shows the four PSD profiles that we obtained:

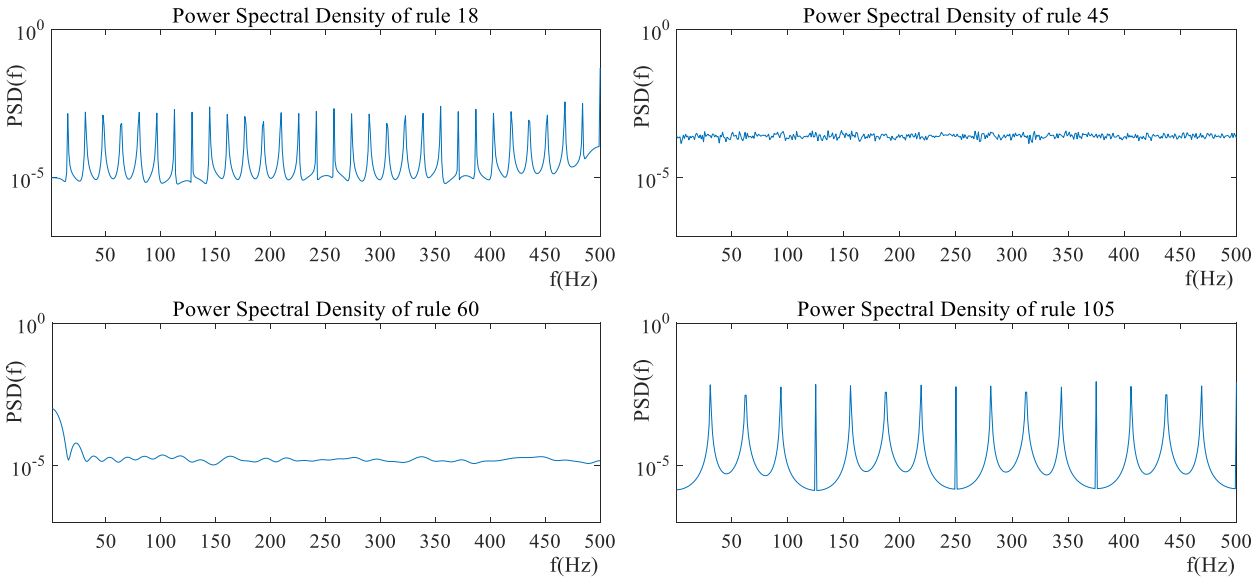

Fig. 10: Power Spectral density of rules 18, 45, 60 and 105.

These graphics have been produced for $t = 1000$ time steps starting from an initial random binary seed with $\mathrm{P} = 0.5$. We chose this amount of steps (and consequently the band width shown in Fig. 10) because a 64-cells ECA can be used to generate a measurement matrix to sample a 32×32 image (which would contain 1024 pixels). Since we are interested in PSD profiles that resemble white noise, from Fig. 10 it is possible to see that this only applies to rules that show an evolution similar to that of rule 45, namely, rules 30, 45, 75, 86, 89, 101, 135 and 149. Rules similar to rule 18 and rule 105 contain far too many repeating patterns while rule 60 shows the presence of repeating patterns at low frequency, which means that two adjacent rows of the generated matrix could potentially have high mutual coherence, hence voiding RIP.

2.3.2 Power Spectral Density of LFSR

We devised a MATLAB experiment in which we evolved LFSR of different lengths over an equal period of time. Since, unlike ECA, LFSR are not easily scalable we were curious to see

how the choice of number of bits would affect their PSD (Fig. 11).

This was also of interest because there are some BCS implementations that, in order to reduce hardware resources, suggest not only to separate the pixel array into small blocks but also to use a small measurement matrix repeated identically in each block instead of partitioning a global measurement matrix shared by the whole array.

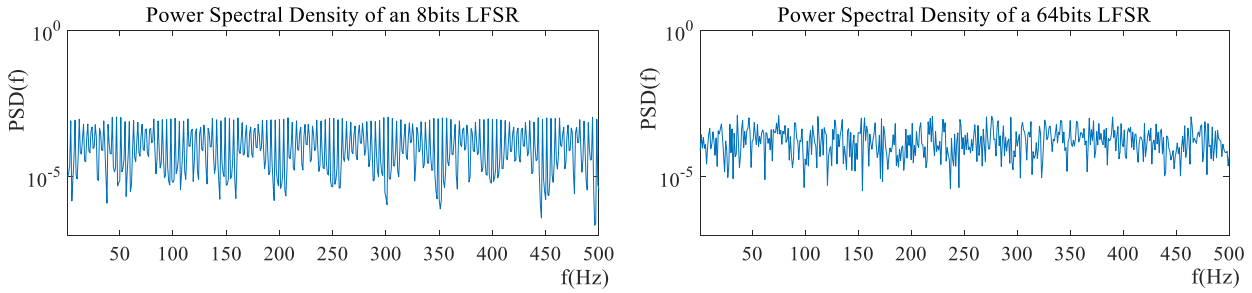

Fig. 11: PSD of LFSR having 8 and 64 bits respectively.

These graphics have been produced for $t = 1000$ time steps starting from initial random binary seeds with $P = 0.5$. From this simulation we can extract two conclusions; the first is that, if the number of compressed samples is tied to the size of the entire image and we used a PRNG that is too small, we might be forced to evolve it too many steps and as such we would incur into repeating patterns as the left-hand graphic of Fig. 11 shows. The second conclusion is that the amplitude of the pattern of a LFSR presents a greater variation than that of ECA as well as some unwanted fluctuations. This difference in behaviour can be explained analysing the mechanism that the different PRNG use to evolve from one state to the next.

2.3.3 THE INFLUENCE OF TEMPORAL EVOLUTION ON PSD AND BOUNDARY CONDITIONS

As all finite discrete systems, all PRNG have a maximum amount of possible states that is tied to the total number of their constitutive elements:

$$T_{max} = M_{max} = 2^{N_f} - 1 \tag{25}$$

It is true that a LFSR whose period exceeds T_{max} repeats the exact same sequence over and over. It is also true that among all possible ECA outputs not all states are equally likely to occur [Wolf94]. It is important to notice though that, given the exponential dependence in Eq. (25), if the number of elements is large enough, the sheer amount of possible combination is sufficient to make these problems negligible but, to complete the study on the dynamic behaviour of PRNG, we must at least try to understand the mechanism that promotes their temporal evolution.

Fig. 7 shows that the condition placed on the evolution of an 8-bit LFSR is imposed through

XOR gates tapping flip-flops 4, 5, 6 and 8. The flip-flops that must be tapped to allow an N_f-bits LFSR to cycle through all $2^{N_f} - 1$ combinations are tied to primitive polynomials with binary coefficients and degree equal to N_f [Ndaw15]. For this reason, if the number of registers varies, then the taps have to be changed as well.

From Fig. 7 it is evident that each step in the temporal evolution of a LFSR is simply formed by displacing the previous step forward by one unit and then adding a new bit at the beginning. As we said in the introduction, a necessary requirement placed upon a measurement matrix is that it respects the RIP, which means that it must be nearly orthonormal, at least when operating on sparse vectors [Cand06]. Orthogonal matrices have uncorrelated columns and rows. For this reason, even though RIP and mutual coherence are two different concepts, a matrix $\mathbf{\Phi}_G$, compliant with RIP, and its transpose $\mathbf{\Phi}_G{}^T$ will both have low mutual coherence by definition [Ober17]. The fact that each step of a LFSR is closely tied with the previous one elevates the mutual coherence of $\mathbf{\Phi}_G{}^T$. In fact, LFSR main purpose is to offer pseudo-random sequences of decimal numbers, the form that these numbers take in the binary domain is an entirely different problem. On the contrary, since the conditions imposed on the evolution of a cell in an ECA are spatially local (Fig. 8), the shifting of data is avoided, so that ECA comprised of many cells do not deteriorate the mutual coherence of $\mathbf{\Phi}_G{}^T$. Moreover, this makes it possible to introduce three different kinds of boundary conditions on the cells at the extremes of the cellular automaton [Wolf83]:

- Periodic: the cells at the extremes are considered neighbours of one another.
- Reflective: the outputs of the cells at the extremes are doubled so that these cells can considered their own neighbour
- Fixed: the boundary values are set a priori from the exterior.

It has been shown [Wolf83] that periodic boundary conditions can be used to emulate a ECA of infinite length since locally, no boundary is apparent to any cell. This periodic boundary condition is most helpful when the system that we are trying to implement tries to mimic a sub-Gaussian distribution to generate rows that have a very low probability of being almost empty or almost full.

2.4 ECA FROM A GEOMETRICAL PERSPECTIVE

Through PSD analysis we have established the superior performance of eight ECA rules over that of LFSR, when it comes to the generation of recursive measurement matrices. At least in terms of their dynamic behaviour, that is, in term of correlation among the outputs of their

adjacent elements and of possible repeating patterns. Remember that this were two of the requirements established to approximate $\mathbf{\Phi}_G$, a binary measurement matrix obtained using a PRNG, to $\mathbf{\Phi}_R$, a binary measurement matrix obtained selecting each element at random from a binary probability distribution. But, among Class III rules, we still have all these possible choices and we have yet to establish their equivalence and/or differences beyond the fact that apparently they share similar power spectra.

When we look at the 256 possible ECA configurations from a geometrical perspective though, we find that, in reality, many of these rules can be considered equivalent under simple transformations. For Class III rules this means that the pattern that they generate will have the same degree of randomness. There are two types of transformations that deliver geometrically similar structures. The first one is the reflection of the pattern generated by a rule using a vertical axis. The rule resulting from this transformation is called a mirrored rule. An example of it is shown in Fig. 12.

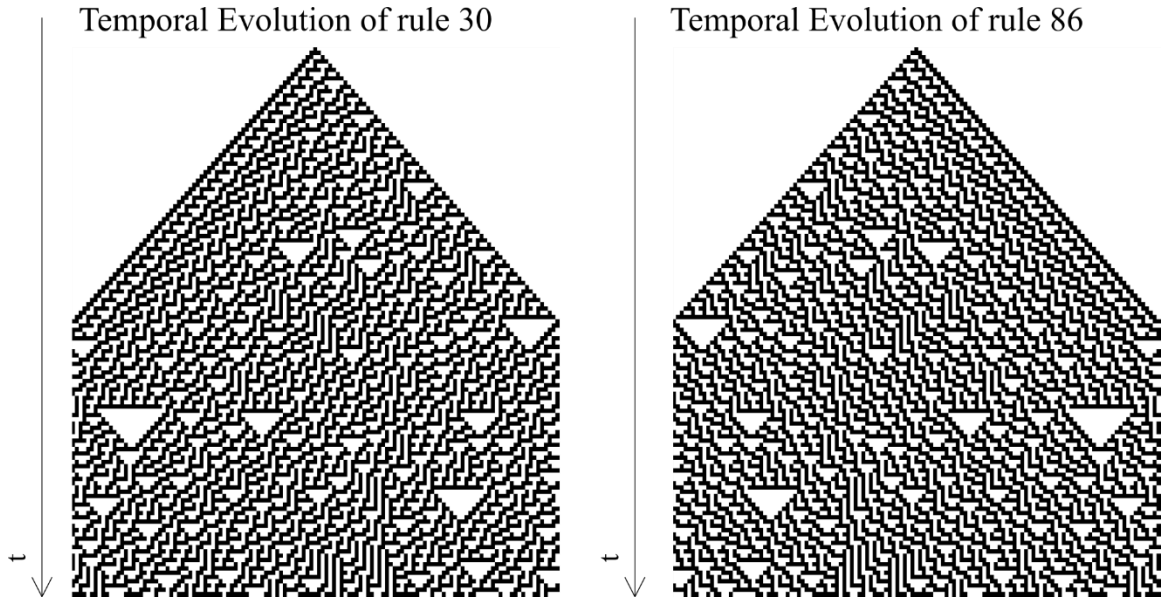

Fig. 12: Temporal evolution of elementary cellular automata following rule 30 and 86.

These patterns have been generated using MATLAB. The top rows in this image represent the seed or first step in the temporal evolution of rules 30 and 86. By assigning the colour black to the elements of the pattern that were ones and white to those that were zeros and letting the seed evolve several steps we obtain Fig. 12. For simplicity, in both cases we have used an initial seed that had only one non-zero element in its middle.

After looking at the results of this transformation, we notice that what we thought were eight separate rules with similar power spectra, are in reality four pairs of mirrored rules and each pair has an identical power spectrum.

The second transformation that delivers geometrically equivalent rules is obtained by exchanging the roles of ones and zeros in the definition of the rule itself. The resulting rule is called the complementary rule. We can see an example of it in Fig. 13.

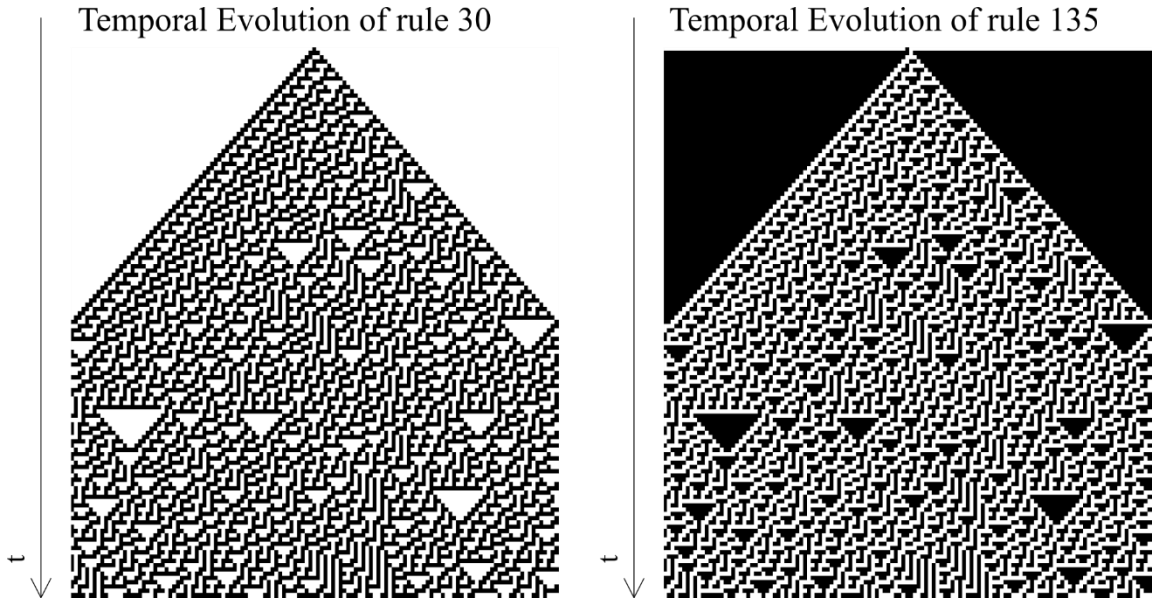

Fig. 13: Temporal evolution of elementary cellular automata following rule 30 and 135.

Once again, both patterns of Fig. 13, just like those of Fig. 12, have been generated through MATLAB starting from seeds that only had one element different from the others in their middle. Whereas the presence of a vertical axis in amphichiral[1] rules introduces problems related to mutual coherence, the inversion of the elements inside the pattern does not. Nonetheless, this second transformation helps us better understand the similarities among our eight choices. What we first thought were eight different choices that then became four are, in reality, only two. In fact, among the eight choices that PSD left us with, rule 30 finds its complementary rule in rule 135, its mirrored rule in rule 86 and its complementary mirrored rule in rule 149. The same can be said for rules 45, 101, 75, and 89, respectively. Note that, even if this analysis does not help us discriminate among our possible choices it still reveals something important; we can reduce our study to rules 30 and 45 without loss of generality.

2.5 Density of the PRNG Output

The last requirement established to approximate $\mathbf{\Phi}_R$ by $\mathbf{\Phi}_G$ was that the number of non-zero elements in a row of $\mathbf{\Phi}_G$ must approximate $P \approx 0.5$. To study the number of non-zero elements in a row of $\mathbf{\Phi}_G$ let us define the density of a PRNG output ($\mathrm{DO}(t)$) as the average of all the states of its binary elements at a given discrete time step *t* (following a notation similar to [Nina08]):

[1] An object is amphichiral if it is superposable with its mirror image.

$$\mathrm{DO}(t) = \frac{1}{N}\sum_{h=1}^{N} \varphi_h(t) \text{ where } t = 1,2,\cdots,M \tag{26}$$

being $\varphi_h(t)$ the state of the *h*-th element of the PRNG at time *t* as well as the *t*-th element of the *h*-th column of $\mathbf{\Phi}_G$, N the number of elements of the PRNG as well as the number of columns in $\mathbf{\Phi}_G$ and M the number of discrete time steps of the PRNG as well as the number of rows in $\mathbf{\Phi}_G$. Let $t = 1$ be the initial condition when the first row of $\mathbf{\Phi}_G$ coincide with the seed that has been loaded in the memory on-chip. Since the elements of $\mathbf{\Phi}_R$ are extracted from a symmetric probability distribution, an optimal seed should have half of its elements set to 0 and half set to 1 and as such $\mathrm{DO}(1) = 0.5$.

To analyse the performance of LFSR and ECA in this respect it is necessary to evaluate how fast their states can reach $\mathrm{DO}(t) = 0.5$ in the eventuality of having a suboptimal initial seed in which the number of elements set to 0 differs from the number of those set to 1. To do so we devise a MATLAB experiment in which we evolve a 64-cells rule-30 ECA, a 64-cells rule-45 ECA and a 64-flip-flops LFSR using 65 different initial seeds having an density that varies from $\mathrm{DO}(1) = 0$ all the way to $\mathrm{DO}(1) = 1$. In the result of the experiment we represent $\mathrm{DO}(t)$ using greyscale elements in which $\mathrm{DO}(t) = 0$ is coloured white and $\mathrm{DO}(t) = 1$ is coloured black.

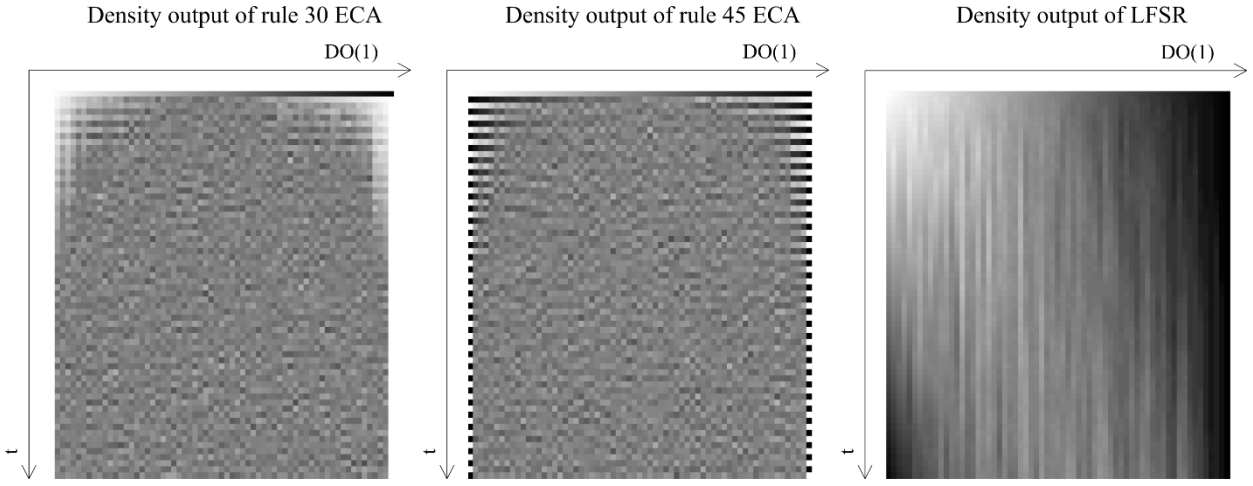

Fig. 14: Temporal evolution of ECA following rules 30 and 45 and of a LFSR.

In Fig. 14 the top line represents these initial seeds with increasing density, from the leftmost $\mathrm{DO}(1) = 0$ to the rightmost $\mathrm{DO}(1) = 1$. Each step underneath shows the temporal evolution of the PRNG output density evolving from each initial seed configuration.

Fig. 14 shows that LFSR density changes in time at a much slower rate than class III ECA. This implies that in the presence of sub-optimal initial seed, the DO of LFSR would take several time steps longer than ECAs to achieve a stable situation centred around $\mathrm{DO}(t) = 0.5$. As reported in [Nish11], the behaviour of LFSR is more similar to class IV ECA, where randomness

in the evolution is linked to the randomness of the initial configuration, rather than class III ECA where the chaotic evolution is introduced by the rule itself. For this reason, in order to approximate $\boldsymbol{\Phi}_R$ using LFSR much care should be placed in the selection of an appropriate initial seed. ECA appear to be a safer choice to generate a $\boldsymbol{\Phi}_G$ rather than LFSR at least in terms of number of distribution of non-zero elements starting from seeds that are not ideal.

Fig. 14 also shows an interesting behaviour for ECA when its seed is either $\mathrm{DO}(1) = 0$ or $\mathrm{DO}(1) = 1$ and for LFSR when its seed is $\mathrm{DO}(1) = 1$. These particular seeds represent the forbidden states of these PRNG. All PRNG have forbidden states. A forbidden state is a state in which the PRNG continuously delivers a fixed output (rule 30 and LFSR) or an oscillating one (rule 45). These states though do not pose any issues in real-world implementations because it is impossible for a PRNG to stumble upon them accidentally. Even though these states do not pose any problem during standard operation, to use PRNG in CS-CIS, it is important to be aware of their presence in order to avoid sending them accidentally. The only particular concern regarding forbidden states happens when power is first applied to a circuit. Since each register can randomly start up containing either a zero or a one, a PRNG could power up containing its forbidden state, but this can be quickly taken care of initializing the PRNG with an opportune seed value.

Once again this analysis presents ECA as a better solution over LFSR when it comes to recursive measurement matrices generation. When we combine these results with those of PSD, it is evident that to design a CS-CIS that is both functional and as close to the ideal case as possible we should select ECA as the choice to make.

Fig. 14 also presents some interesting differences in the behaviour of rules 30 and 45, differences that become especially evident the more the initial seed is removed from the central position (i.e. from a symmetric probability distribution of the seed). We can see that while rule 30 steadily evolves from a low density towards equilibrium, rule 45 oscillates between steps of low density and steps of high density. This behaviour can be intuitively understood by looking at the evolution patterns that these two rules impose on the single cells Fig. 9 and Fig. 15:

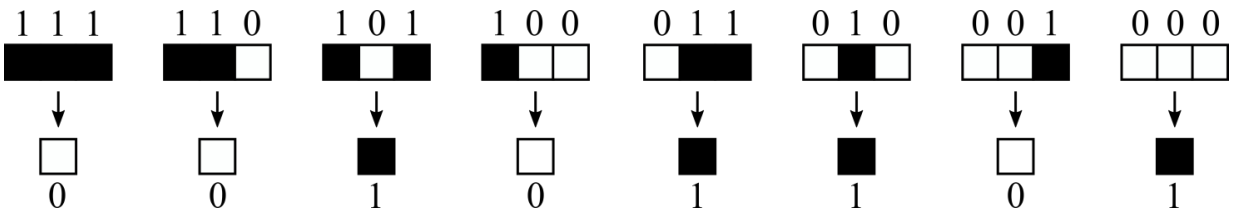

Fig. 15: Evolution pattern of a cell of an elementary cellular automaton implementing rule 45.

The more the density of the initial seed is unbalanced towards a situation where many cells

start from the same initial state, the more the probability of having three equal inputs for every cell increases. In other words, an unbalanced initial seed favours one of the combinations at the extremities of Fig. 9 and Fig. 15 but, whereas in rule 30 these combinations both evolve into the same null output, in rule 45 they lead to an oscillating behaviour where an initial null state evolves to a full one in the next step and vice versa. Even if, as shown in Fig. 14, these oscillations are rapidly extinguished by the random behaviour of the rule, they are still undesirable because, when implemented in a CMOS circuit they will raise the overall power consumption of the CS-CIS. For this reason, at least when it comes to design a measurement matrix in CMOS, rule 30 seems to be a more solid choice over rule 45. Given the geometrical equivalences presented in section 2.4, rules 101, 75, and 89 share the same behaviour and therefore can be excluded as well.

2.6 CIRCUITAL EQUIVALENCE OF RULES 30, 86, 135 AND 149

One last criterion to choose which among our remaining rules is better suited for CMOS implementation is the number of transistors needed to describe the functionality of a single cell. The lower this number is, the more compact its design will be. Fig. 16, Fig. 17, Fig. 18 and Fig. 19 represent the minimum amount of logic gates that are needed to describe these rules:

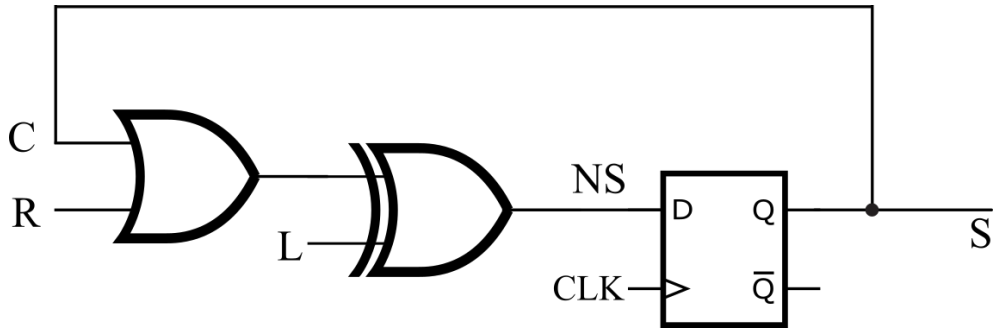

Fig. 16: ECA cell implementation of Rule 30.

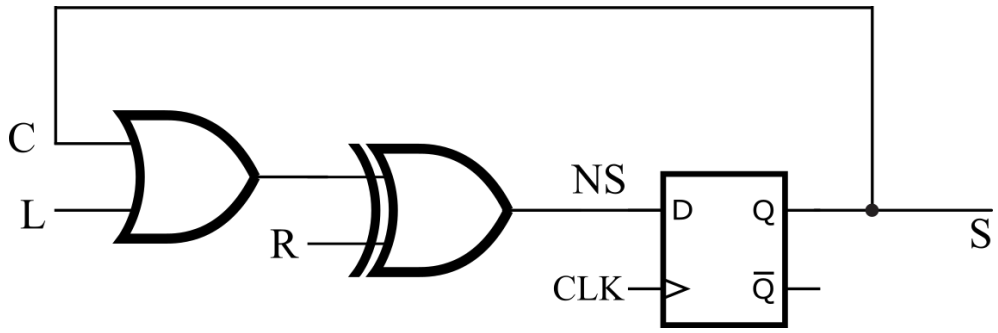

Fig. 17: ECA cell implementation of Rule 86.

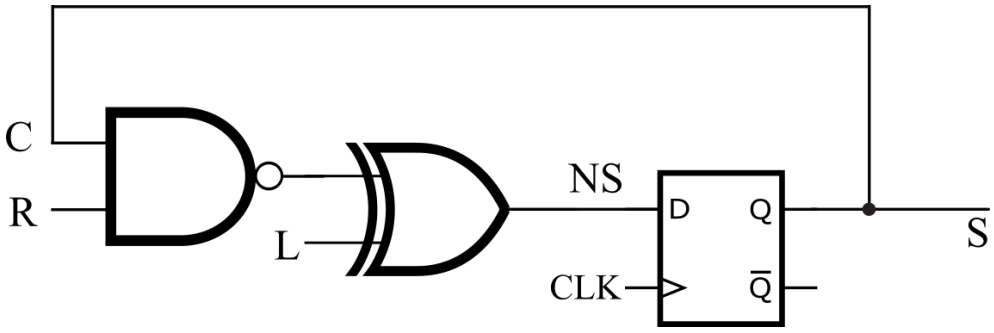

Fig. 18: ECA cell implementation of Rule 135.

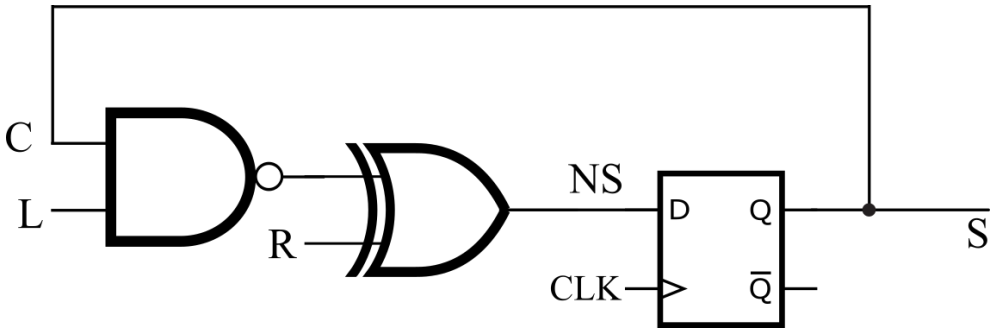

Fig. 19: ECA cell implementation of Rule 149.

In each figure above we have defined as S the actual state of the cell that will be used in the pixel array to generate a compressed sample and as NS the next state, loaded at the flip-flop input and ready to be sent at the next clock cycle and that corresponds to a new row of the recursive measurement matrix. C is the input that copies the actual state of the cell whereas L and R are the actual states of its two closest neighbours, to the left and to the right respectively. The first thing that is possible to notice is the striking resemblance of these implementations; each rule can be described using two logic gates and the logic gate that connects each rule to the flip-flop in charge of controlling the time evolution of the ECA is always an XOR (Fig. 20).

In hindsight these similarities were not unexpected; in fact, they are a simple reflexion of the geometrical analysis performed in section 2.4. If we were to pair each rule with its mirrored rule, i.e. rule 30 (Fig. 16) and rule 86 (Fig. 17), it is clear that one can be converted into another simply by inverting inputs L and R. Similarly, if we were to pair each rule with its complementary rule, i.e. rule 30 (Fig. 16) and rule 135 (Fig. 18), it is possible to see that, since an OR is simply a NAND whose input has been inverted, it make sense that the two images presented in Fig. 13 looked the same with inverted colours.

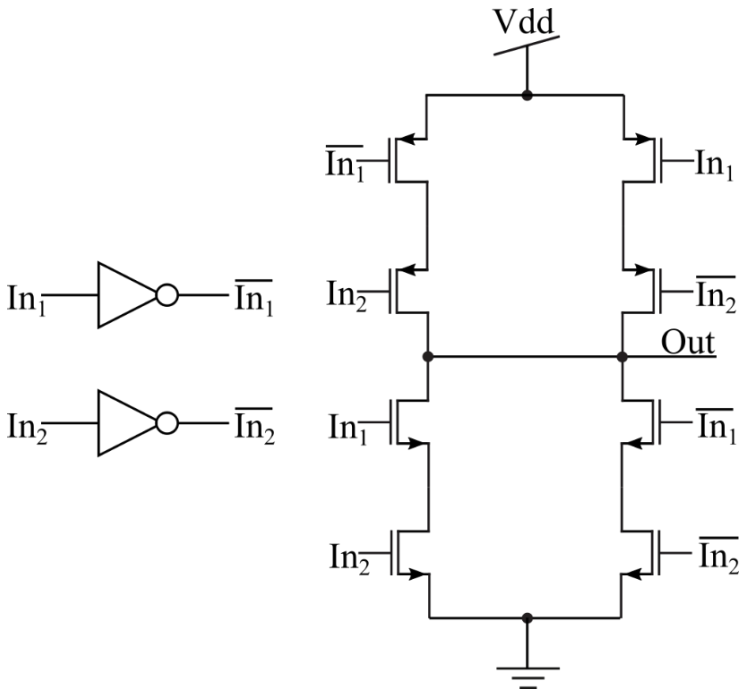

Fig. 20: Basic XOR diagram.

Even the XOR that they all share (Fig. 20) and that has 50% of its possible outcomes equal to zero and 50% equal to one, can be seen as a consequence of the choice that we made when we selected rules whose eight possible combinations (Fig. 9) were equally divided among ones and zeros. More importantly, after a little manipulation, it is possible to see that, the extra inverter that is present in the OR of rules 30 and 86 can be incorporated into the XOR that follows and that, as a result, all of these circuits need the same amount of transistors in their designs.

To design our circuit we will choose rule 30 but we have proved that each of the four alternatives here presented would be mathematically and electrically equivalent in all aspects for both CS requirements and electronic design.

CHAPTER 3
COMPRESSED SAMPLES IN THE DIGITAL DOMAIN

By now it should be obvious that CS pixel arrays, given their nature, are not meant to be read in a row by row fashion. They are instead designed to deliver linear combinations of the pixels values. As such, the amount of bits required to correctly convert these combinations from analog to digital signals is much higher than the 8 to 10 bits usually required in standard CMOS image sensors (CIS) Eq. (11). For convenience, it is opportune to remember that, for binary measurement matrices, the number of bits needed to convert a compressed sample, is proportional to the size of the overall pixel array and to the number of bits with which we intend to recover the value of each pixel output during reconstruction.

In our case we plan to design a 64×64 pixels array and we would like to maintain an 8 bits resolution per pixel value, and as such we will need a 20 bits A/D converter. Note that, since the amount of pixels in the array is fixed, if we were to reduce the number of bits of the conversion, we would be compromising reconstruction by losing bits on each pixel value of the recovered image.

3.1 LIMITATIONS ON THE DYNAMIC RANGE OF COMPRESSED SAMPLES IN A/D CONVERSION

Imagers employ ADCs to map the analog pixel values onto digitally-encoded values. Such mapping process involves time-discretization and amplitude-discretization thereby leading to frequency limitations and errors. Handling these limitations and errors might require dedicated ADC architectures to preclude unnecessary power and area consumption [Leñe18].

Typical ADC architectures choices for image sensors are: pipeline ADC, slope ADC, cyclic ADC, successive approximations ADC and Σ-Δ ADC. Independently of the chosen architecture, there are three non-ideal aspects on the characteristics of a real ADC that can be expressed as uncertainties that limit the precision with which the variation of a continuous signal can be correctly represented in a sampling process. This, in turn, limits the effective number of bits of

the resulting conversion [Malo11].

Uncertainties can be expressed in the form of noise, a good indicator of the effect that a certain level of noise has on a signal is given by the signal to noise ratio (SNR), which is defined as the ratio between the powers of the desired signal and of the noise that affects it. The first limitation common to all ADC is the sampling-time jitter. This error is introduced by the non-ideality affecting the circuit's clock periodicity and, to some degree, by propagation delays between the controls that initiate the sampling phase and the circuits that perform the actual sampling that can be known only to a certain extent. The jitter error $\epsilon_j(nT)$[1] introduced on the sampled signal x_o by the sampling time uncertainty $\delta(t)$, for the n-th sample in a sampling process of period T can be modelled linearly using a Taylor series truncated at the first order:

$$\mathrm{x}_o(nT) = \mathrm{x}_i(nT) + \epsilon_j(nT) = \mathrm{x}_i(nT) + \frac{d\mathrm{x}_i(t)}{dt}\Big|_{nT} \delta(nT) \tag{27}$$

Following Eq. (27), the sampling time jitter can be seen as linear noise that depends on the rate of change in time of the ADC input signal $\mathrm{x}_i(t)$. The sampling uncertainty is usually modelled as white noise, which is uniformly spread across the whole frequency spectrum. The SNR of an ADC is usually computed by feeding a series of sine waves as its inputs. If $\mathrm{x}_i(t)$ took the form of a sine wave of signal full scale X_{FS} and frequency f_i, the power associated to the average sampling time jitter $\bar{\delta}$ would take the form:

$$P_j = \frac{1}{2}\left(\mathrm{X}_{FS}\pi f_i \bar{\delta}\right)^2 \tag{28}$$

The second uncertainty is introduced by the quantisation process and it is due to the rounding error $\boldsymbol{\epsilon}_q$ between the continuous input voltage to the ADC and its digitised output. This error can be modelled as linear noise that diminishes with the increase of B, number of bits of the converter. Considering an ADC with range X_{FS} and B bits, the sampled signal $\mathbf{x}_o$ can take one of 2^{B} possible discrete values:

$$\mathrm{x}_{o\,m} = \mathrm{x}_i + \epsilon_q = (2^m + \tfrac{1}{2})\frac{\mathrm{X}_{FS}}{2^{\mathrm{B}}} \quad m \in [1, \mathrm{B}] \tag{29}$$

This is a direct consequence of the fact that the value of the input signal x_i can only fall in one of the intervals of size

$$\Delta = \frac{\mathrm{X}_{FS}}{2^{\mathrm{B}}} = 1\mathrm{LSB} \tag{30}$$

[1] the jitter error is defined with the subscript 'j' of *jitter* and not 'i' of *input* to remind the reader that it is an intrinsic characteristic of the converter even when it can be modelled with a truncated Taylor series that makes it proportional to the first order derivative of the input signal.

that the quantiser has at its disposal. Δ is known as the quantisation step. The probability distribution of the quantisation error of a sample can be modelled as uniform on the corresponding interval and zero in all others. The density of this uniform distribution is the inverse of the quantisation step itself. Once again we have to express this uncertainty in the form of noise and the power associated to the quantisation noise can be computed as:

$$P_q = \int_{-\Delta/2}^{+\Delta/2} \frac{\epsilon_q^2}{\Delta} d\epsilon_q = \frac{\Delta^2}{12} \tag{31}$$

The last uncertainty that affects all ADC is associated to the sampling switch; it depends on the thermodynamic fluctuations of the amount of charge that the sampling capacitance holds. This noise can be modelled as a thermal noise whose power can be expressed as:

$$P_T = \frac{k\Theta}{C} \tag{32}$$

being k the Boltzmann constant, Θ the temperature of the ADC expressed in kelvin and C its capacitance. It is possible to see that this third uncertainty, instead of being derived by the type of operations that an ADC carries out, it only depends on the physical characteristics of the circuit itself. The only way to completely remove it would be by bringing the temperature of the circuit to absolute zero or by having infinite sampling capacitance.

Once we have derived an expression for the noise power associated to the different sources of uncertainty that befall on a real ADC, we can use SNR, expressed in decibels, to estimate the equivalent number of bits (ENOB) that these uncertainties allow. Considering once again as input signal a sine wave of signal full scale X_{FS} and frequency f_i. We can derive the power of such input as:

$$P_{in} = \frac{1}{T}\int_0^T \frac{X_{FS}^2}{4}\sin^2(2\pi f_i t)\, dt = \frac{X_{FS}^2}{8} = \frac{\left(2^B\Delta\right)^2}{8} \tag{33}$$

Joining Eq. (28), Eq. (31), Eq. (32) and Eq. (33) we can estimate ENOB as:

$$\text{ENOB} = \frac{\text{SNR}_{dB} - 1.76}{6.02} = \frac{10\log_{10}\left(\frac{P_{in}}{P_j + P_q + P_T}\right) - 1.78}{6.02} \tag{34}$$

To understand how these sources of errors affect the design of and ADC we can plot ENOB over the number of bits specified in the design of an ADC:

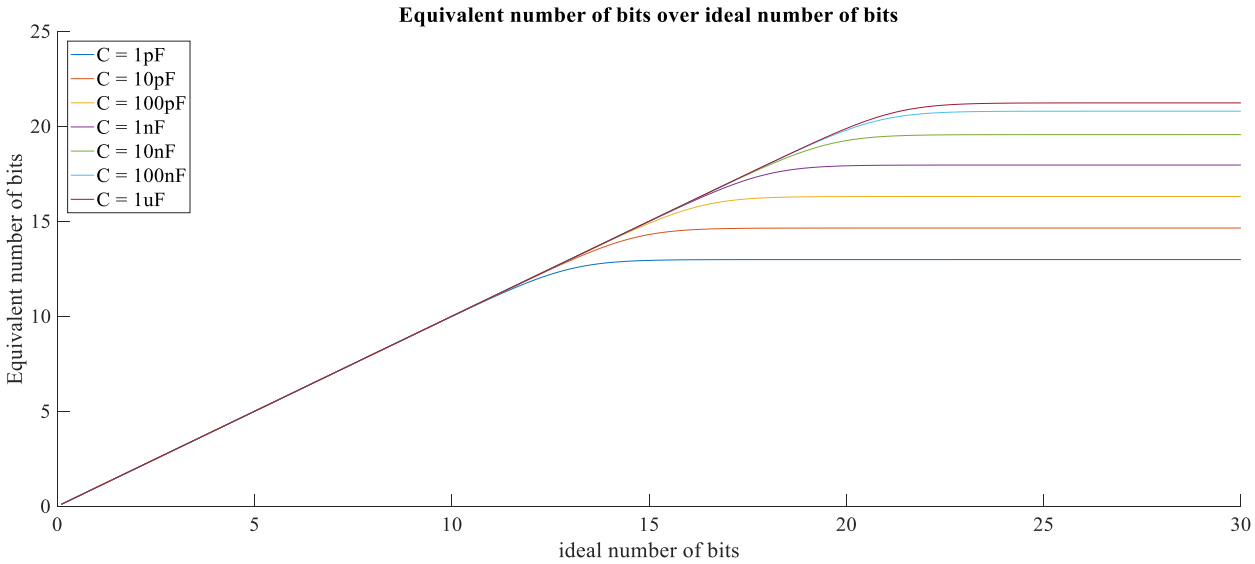

Fig. 21: Estimated number of bit over number of bits specified in the design.

We can see from Fig. 21 that, when the number of bits in the design of an ADC is low, the quantisation error dominates the SNR and as such the ENOB increases linearly with its reduction. As the demand for precision surpasses a certain limit the thermal noise first and the jitter noise second become dominant creating an asymptote above which ENOB cannot go. Inverting Eq. (34) it is possible to obtain the value of SNR associated to the 20 bits that our ADC would need to digitize the compressed samples:

$$\mathrm{SNR_{dB}} = (6.02\mathrm{ENOB} + 1.78) \approx 122\mathrm{dB} \tag{35}$$

To give a first order approximation of the design parameters that are needed to achieve 122dB of SNR we will handle each error source independently from the others as if it was the only one affecting the ADC. To design our sensor we have chosen to use UMC 180nm technology. We will have $\mathrm{X}_{FS} = 1.8\mathrm{V}$. As the quantisation error decreases with the increase in number of bits, we will focus only on sampling time jitter and thermal noise. We need to raise the asymptote in Fig. 21 above 20 bits.

As seen in Eq. (28), the error associated to the sampling time jitter depends on the frequency of the signal that needs to be converted. Given the fact that pixel values are encoded by 8 bits and compressed samples by 20 bits, the maximum subrate, S_{max}, at which we can compressively sample an image before the overall size of the collected compressed samples surpasses that of the uncompressed image is 0.4. As a design parameter we would like our sensor to be able to collect as many as 30 frames per second, N_f. The maximum amount of samples N_{CS} that our imager must be able to take each second is:

$$N_{CS} = (64 \times 64) \times S_{max} \times N_f \approx 50\,kS/s \tag{36}$$

According to the Nyquist Theorem (remember that CS in CS-CIS applies to spatial sampling and that analog to digital conversion still follows the same old rules), it is necessary to sample twice as fast as the highest frequency that we want to measure. As such, the anti-aliasing filter will allow a maximum input frequency of 25 kHz. If we were to consider only sample time jitter as source of noise, combining Eq. (28) and Eq. (33):

$$\mathrm{SNR_{dB}} = 10\log_{10}\frac{P_{in}}{P_j} = 10\log_{10}\frac{1}{4\left(\pi f_i \bar{\delta}\right)^2} \tag{37}$$

From which we derive $\bar{\delta} = 2 \times 10^{-18} S = 2as$ for an SNR of 122dB. Combining Eq. (32) and Eq. (33) we can obtain an expression for the thermal noise:

$$\mathrm{SNR_{dB}} = 10\log_{10}\frac{P_{in}}{P_T} = 10\log_{10}\frac{X_{FS}^2 C}{8k\Theta} \tag{38}$$

Considering an ADC working temperature of 300K, we would need $C = 16\mathrm{nF}$ of sampling capacitance. In UMC 180nm, the biggest capacitor can be implemented using metal 5 and metal 6; it has a capacitance of 10.03pF and occupies an area of 0.1 mm^2. To achieve the required capacitances we would need to stack in parallel approximately 1600 metal 5 and metal 6 capacitors, covering an area of 16 cm^2, which is an impossible size for a microcircuit. For 20 bits ADC, both sampling time jitter and thermal noise lead to design parameters that are beyond UMC 180nm reach.

3.2 TIME TO DIGITAL CONVERSION OF THE COMPRESSED SAMPLES

An SNR of 122dB is a demanding requirement for any analog design, whatever the encoding and technology. As such, a representation in the voltage domain of the compressed samples might be difficult to digitise with enough accuracy. To overcome this obstacle we will try to modulate the analog outputs of the pixels in order to be able to achieve the desired amount of equivalent bits during conversion. Two of the most common CMOS solutions for pixel output modulation are pulse width modulation (PWM) [Klei01], [Kitc05] and pulse frequency modulation (PFM) [Culu03], [Wang06]. Pulse modulation (PM) improves the dynamic range and SNR on each pixel [Chen11]. Both solutions require an in-pixel comparator that triggers an event when the voltage of the photodiode drops below a reference voltage that is set as an outside parameter.

In PWM the output of the operational amplifier will enable the content of a global counter

located outside the pixels array to be stored into in-pixel memories. When a large number of bits are required, the area used for routing the global counter output to each pixel memory is high. In PFM time counting is made in-pixel and uses a self-reset mechanism to recharge the junction capacitor of the photodiode each time the voltage of the photodiode drops below the reference voltage.

3.2.1 Asynchronous Pulse Frequency Modulation

In CS, knowledge about which pixels take part in a sample is encoded inside the measurement matrix and as such, there is no need to store the amount of pulses generated by a single pixel in a dedicated in-pixel memory. These pulses can be sent out of the pixels array as soon as they are generated. The resulting pixel architecture can be exemplified as shown in Fig. 22.

A photodiode can be modelled as a photo-controlled current source in parallel with an ideal diode and a capacitor. The capacitor represents the junction capacitance and the current source represents the current generated by the incident light. When the photodiode is reverse-biased, the current source behaviour is highly linear with respect to incident light intensity. An accurate photodiode model contains a parallel resistance to model the slope of the current-voltage curve at the origin and a series resistance to model the resistance of the contacts. Ideally, the parallel shunt resistance should be infinite and the series contact resistance should be zero. For this reason to model the behaviour of the voltage drop at the cathode of a reversely-biased photodiode, we will only consider the capacitor. For a constant photo-generated current I_{ph}, the voltage drop takes the form:

$$\frac{I_{ph}}{C} = \frac{dV(t)}{dt} \tag{39}$$

being C the junction capacitance. From Eq. (39) it is possible to see that, if the luminosity is constant, than the drop will also be constant. We can integrate this expression and obtain:

$$\int_0^T dt = \int_{V_{ref}}^{V_{rst}} \frac{C}{I_{ph}} dV \tag{40}$$

Eq. (40) links passing of time, T, to the junction capacitance, to the voltage drop between the reset voltage V_{rst} and the reference voltage of the operational amplifier V_{ref} and to the current across the photodiode which in turn depends on the incident light intensity. The output of the operational amplifier is fed back into M_{rst} gate, the PMOS reset transistor. During integration time a pulse train is created so that the frequency, f, of the pulses varies in accordance with the instantaneous of the photo-generated current. The amplitude and width of said pulses is kept

constant. The modulating signal is the photo-generated current intensity flowing through the illuminated photodiode:

$$f = \frac{1}{T} = \frac{I_{ph}}{C(V_{rst} - V_{ref})} \tag{41}$$

To form a compressed sample, the train of pulses is sent to one of two column buses that gather the pulses coming from all the pixels belonging to the same column Fig. 22. The selection signals generated by the cells of the ECA along with the in-pixel logic (XOR) will determine which bus the pixels output will activate. We aggregate these outputs using up-down counters for each column of the pixel array.

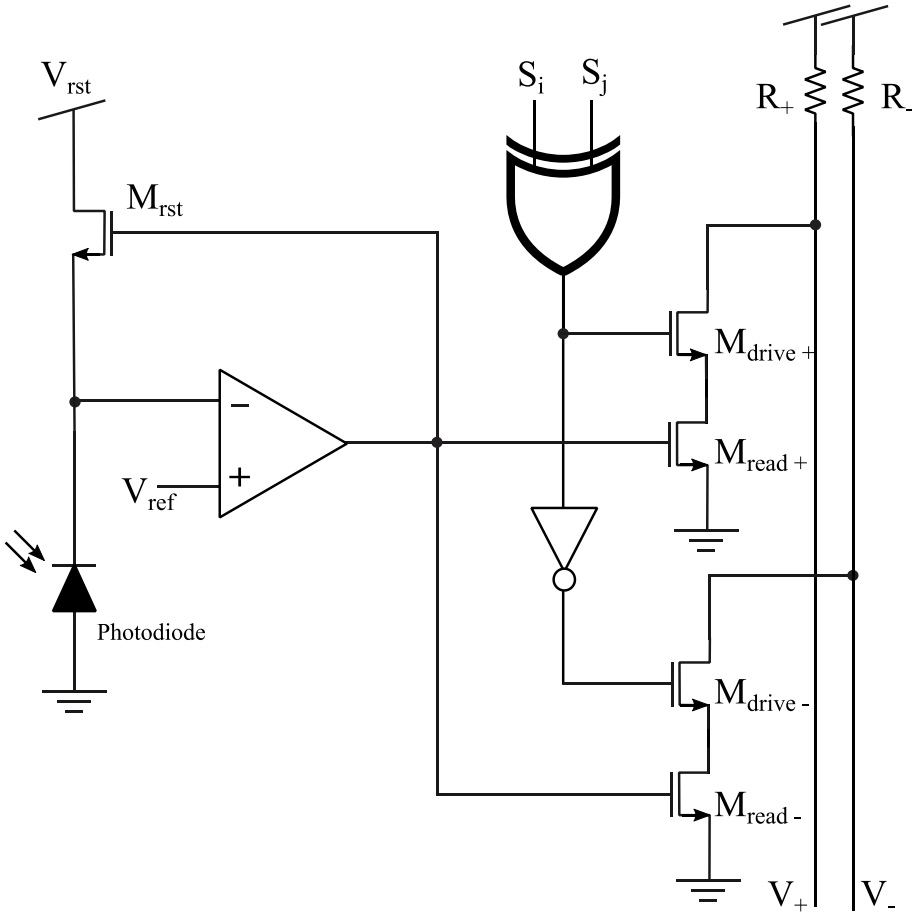

Fig. 22: PFM pixel architecture for CS-CIS.

The use of a varying voltage reference allows us to adapt the frequency of the pulse train for a particular light intensity and avoid saturation of the counters used to collect the compressed samples. Up-down counters allow the generation of differential compressed samples directly in the digital domain where it is easier to improve the required dynamic range. This is equivalent of using a pseudo-random measurement matrix whose coefficient can be either ones or minus ones. The overall architecture of a CS-CIS that incorporates PFM and per column up-down counters is

exemplified in Fig. 23.

By using PFM at pixel level and then counting the pulses at column level we are conveying the problem of the dynamic range of the aggregated signal to the digital domain where it can be increased just incorporating more resources. Additionally, by taking into account the saturation of the counters, bad samples can be discriminated, avoiding reconstruction artefacts [Trev17].

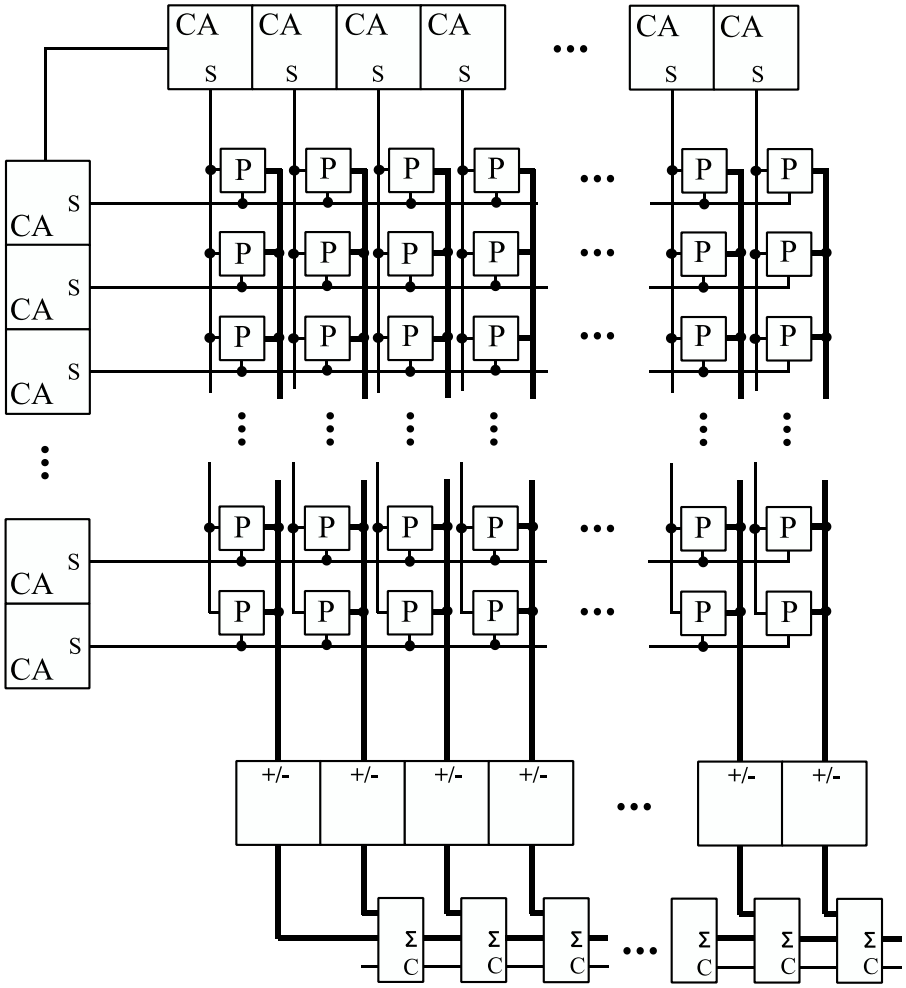

Fig. 23: CS-CIS conceptual floor plan with PFM pixels outputs and up-down counters.

This solution helps to circumvent the uncertainties that are present in all ADC but its implementation in a given technology is still subject to limitations of another nature. Namely, there is a limit on how fast a transistor can be switched on and off. This implies that, the pulses generated by the PFM pixels must be spaced enough and must have long enough duration to allow the counter to count them properly. This limit depends on the technology used, as well as on the characteristics of the driving signal, on the power supply, and on the load. We measured experimentally, for UMC 180nm, how fast a PMOS transistors source can switch from GND to V_{dd} after its gate is set to GND and the time registered for the operation to be completed was around $T_T = 0.7\text{ns}$.

From Eq. (36) we already established that we would like to obtain 50 kS/s. Since we would like to use an up-down counter per column, each counter will possibly receive pulses from a maximum of $N_c = 64$ pixels. Since pixel selection is pseudo-random N_c could be lower but we need to consider the worst case scenario to establish the design feasibility. We would like to describe the output of a pixel with 8 bits and, as such, each pixel will need to pulsate at most $N_p = 256$ times per sample. Combining this information we are able to compute the minimum amount of allocated time to the single pulse for this worst case scenario, which corresponds to when the counter has to operate as fast as possible:

$$T_{PFM_{min}} = \frac{1}{N_p N_c N_{CS}} = 1.22\text{ns} \tag{42}$$

Given the fact that $T_{PFM_{min}}$ is less than twice T_T, even if this solution theoretically allows us to reach 20 bits resolution on the digitalization of the compressed samples, UMC 180nm is not suited to implement it.

3.2.2 ASYNCHRONOUS PULSE WIDTH MODULATION

In standard acquisition techniques, whenever PM of pixels outputs is applied, PFM is usually preferred over PWM as the latter is affected by three major drawbacks [Chen08]:

- A high speed clock is needed to drive the global counter.
- The delay from the global counter to each pixel may introduce error on the final value stored in the in-pixel memory.
- The resolution of PWM is limited by the area required for routing data from the global counter to each pixel.

As we mentioned previously, since the information about which pixels take part in the generation of a compressed sample is encoded in the measurement matrix, there is no need for in-pixels memories. For PWM this means that it is not necessary to send the information contained in the global counter to each pixel of the array. It is possible to send the event triggered by the operational amplifiers of the pixels outside the array to a memory common to a whole pixels column. This consideration is actually beneficial to PWM because on one hand it negates PWM drawbacks and on the other hand it diminishes the frequency at which the circuit needs to operate with respect to PFM. The conceptual floor plan of a circuit that implement PWM without in-pixels memories is exemplified in Fig. 24.

The columnwise up-down counters needed for PFM are replaced with columnwise sample & accumulate (S&A) memories that sample a global counter designed to count backwards from

256 to 0. At the beginning of the integration time, by means of an external signal applied to the pixels reset transistor M_{rst}, the junction capacitances of the photodiodes are charged (Fig. 25). As soon as the voltage at the cathode of the photodiode drops below the reference voltage V_{ref} an event is triggered and sent to the corresponding S&A. The event is terminated by yet another external signal referred as Q in Fig. 25.

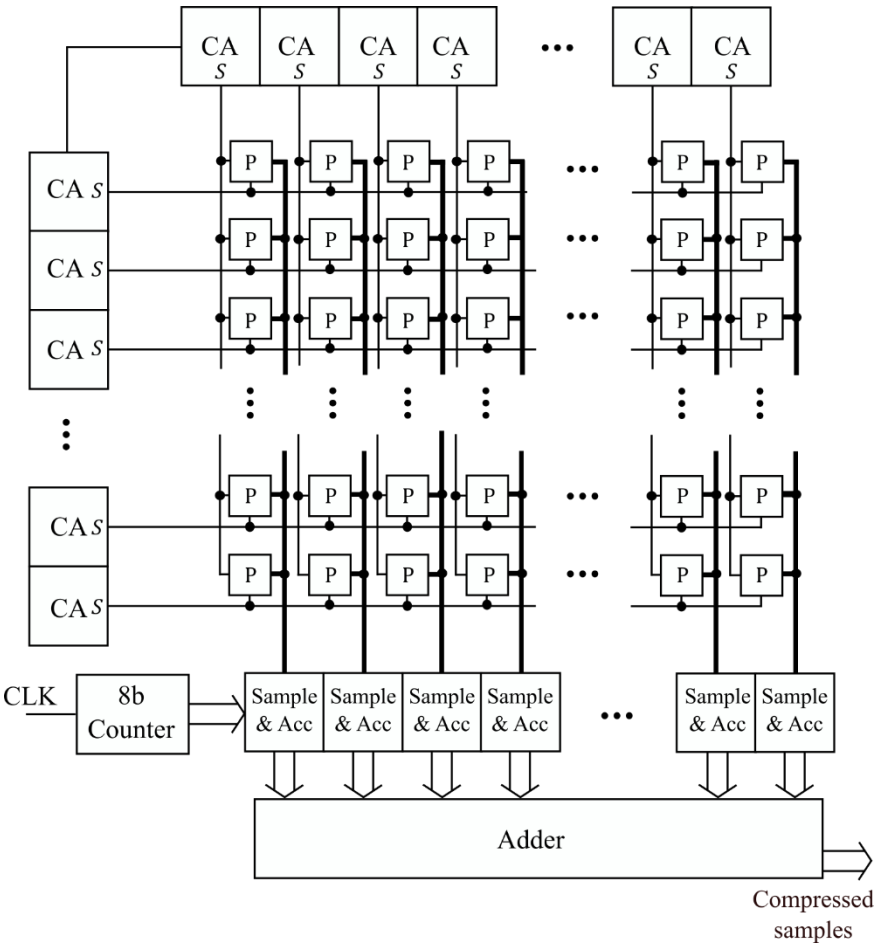

Fig. 24: CS-CIS conceptual floor plan with PWM pixels outputs and Sample & Accumulate.

The external termination signal Q adds a degree of freedom to the pixel control; through Q it is possible to control the duration of the event emitted by the pixels. These events take the form of pulses with controllable duration that are produced asynchronously. The lower the light intensity is, the longer it will take for the pixel to emit its pulse. The later a pulse is sent to de S&A, the smaller will be the value sampled from the counter.

In PWM each pixel sends only one pulse. The burden of encoding the pixel output using the appropriate amount of bits is left to the global counter. Using PWM instead of PFM reduces the amount of pulses sent by each pixel in the array by a factor of 2^8. As such, the maximum frequency at which the S&A needs to operate with respect to the up-down counters is reduced by the same factor.

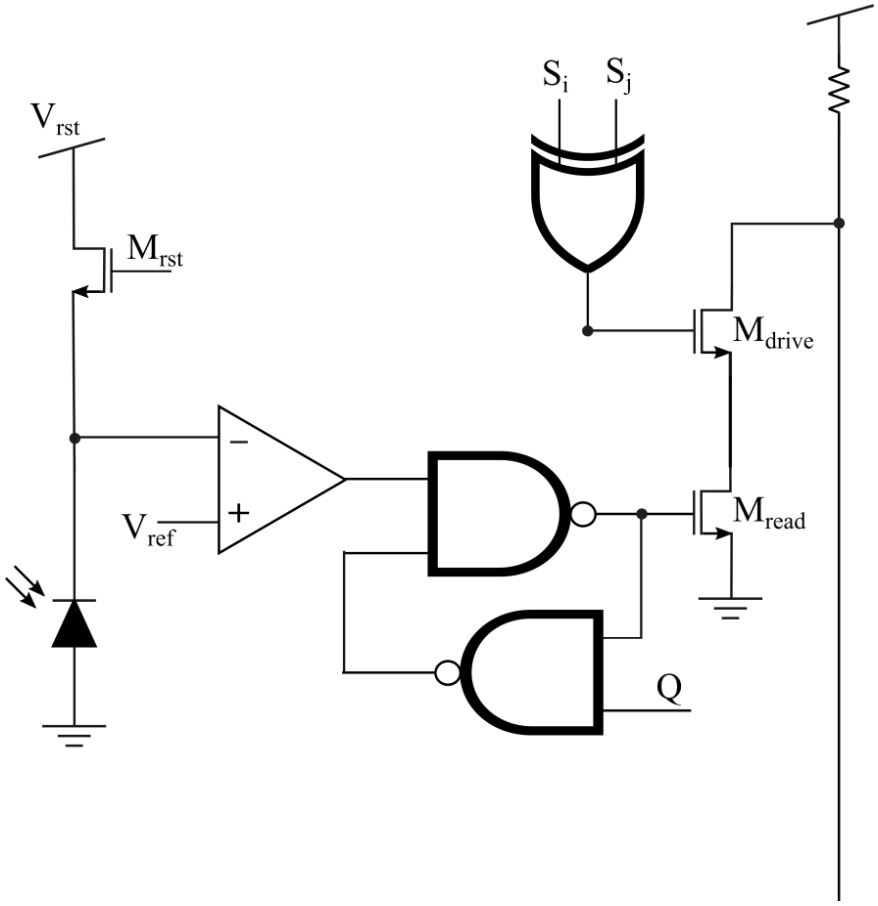

Fig. 25: PWM pixel architecture for CS-CIS.

Eq. (42) can be rewritten as:

$$T_{PWM_{min}} = \frac{1}{N_c N_{CS}} = 312.5ns \tag{43}$$

This result makes this solution implementable in UMC 180nm but it needs an extra precaution: as for PFM, in PWM the voltage drop at the cathode of the photodiode can be modelled with the discharge of a capacitor. We can rewrite Eq. (40) as:

$$I_{ph}T = C(\mathrm{V_{rst}} - \mathrm{V_{ref}}) \tag{44}$$

Eq. (44) states that, for a fixed pair of reset voltage and reference voltage, the product between the photo-generated current and the timing of the generated event is constant. In other words, the equation that unites light intensity to time is hyperbolic. As such, PWM cannot work with a counter that counts at a fixed frequency. The frequency of the counter must change linearly in time, with an expression like:

$$f(t) = f_0 - kt \tag{45}$$

being k a positive constant. Combining Eq. (41) and Eq. (45) and considering the counter counts backword we achieve:

$$\frac{I_{ph}}{C(\mathrm{V_{rst}}-\mathrm{V_{ref}})} - f_0 = kT \tag{46}$$

Using a counter with a linearly changing frequency gives a direct proportionality between the photo-generated current and the timing of an event.

CHAPTER 4
PROTOTYPE ARCHITECTURE AND TESTING

The central element of the CI-CIS architecture (Fig. 26) that we have designed is an array of 64×64 pixels (P). The peripheral circuits implement two functionalities: pseudo-random column and row pixel selection and time-to-digital conversion and addition of pixel values. In the floorplan presented in Fig. 26 the left and top sides of the array are surrounded by a rule-30 ECA that forms a ring of 128 cells (CA) with periodic boundary conditions (see section 2.3.3).

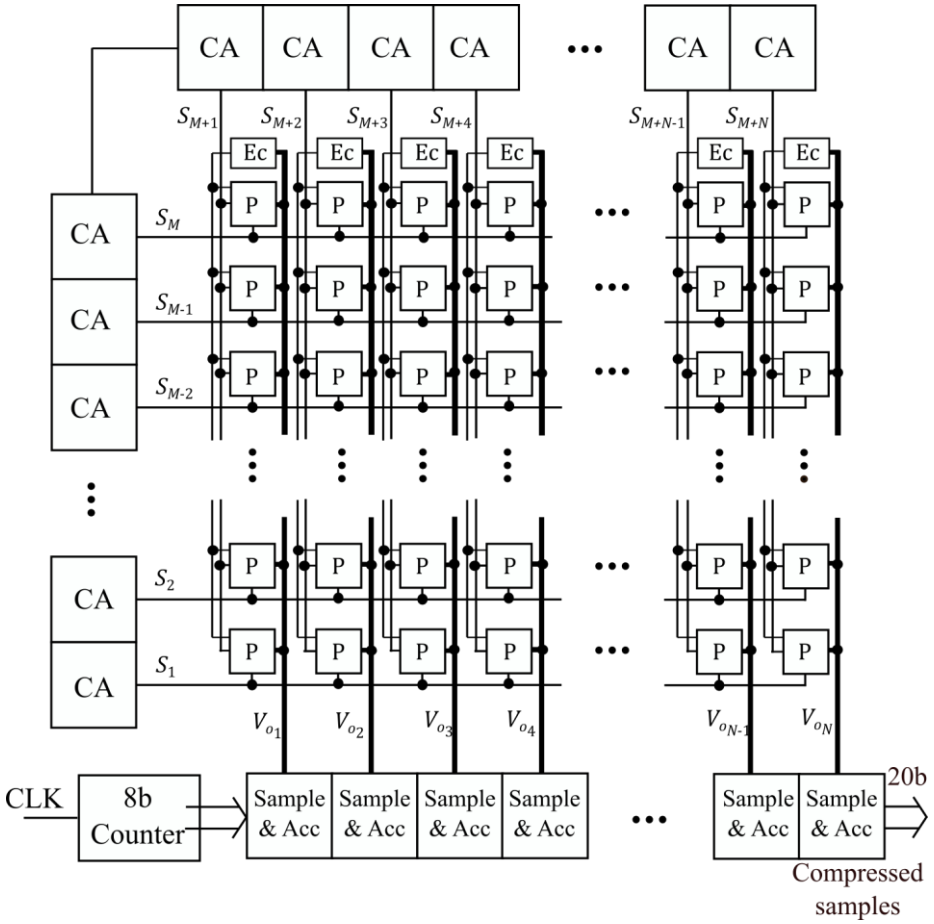

Fig. 26: Circuit Floorplan.

Each cell is used as pseudo-random row/column selector for the array. The pixels are pulse width modulated (see section 3.2.2) and the events that they generate are transmitted through

column busses to 20-bit sample & accumulate elements (S&A) placed at the bottom side.

These pulses encode the pixel value in the period of time that has passed between the pixels reset and the pulse arrival to the S&A. A straightforward method to translate all these times into digital codes is to use the incoming pulses to activate the sampling of a global time counter. This counter is activated by a signal that is sent shortly after the pixel reset. The small delay between pixel reset and counter activation does not pose a problem because the nonlinearity of the characteristic curve of light intensity over photodiode discharge time (Eq. (44)) which (as explained in section 3.2.2) is hyperbolic, makes it impossible for pixels to send pulses immediately upon reset unless the intensity of the incident light approached infinity. Each time a pixel activation pulse arrives at the S&A., the 8 bits of the counter are sampled and added to the sum already stored. After 256 clock periods, the pixel values of each column have been accumulated in 14-bit words, as this is the amount of bits resulting from adding up to 64 8-bit pixel values. After that, the 64 column sums are added together into a compressed sample of 20 bits.

Compressed samples need to be encoded in large digital words; therefore there is an amount of compressed samples beyond which it is better to just deliver the uncompressed image. In our case, as pixel values are encoded by 8 bits and compressed samples by 20 bits, the subrate (S), i. e. the number of samples delivered divided by the total number of pixels in the image, needs to be below 0.4. This means that for a $M \times N$-pixel image, we will always be considering less than $0.4MN$ compressed samples. In addition, as compressed samples are generated sequentially, it is necessary to operate the imager at a frequency (f_{cs}) that is at maximum 0.4 MN times the desired image delivery speed or frame rate (f_s):

$$f_{cs} = S \cdot MN f_s \tag{47}$$

For frame rate $f_s = 30$fps, a sampling subrate $S = 0.4$ and an image of 64×64 pixels, compressed samples can be generated at maximum frequency $f_{cs} \approx 50$kHz, that is 20μs per compressed sample. But, images, such as natural images or MRIs, are commonly piecewise smooth [Sha012], Therefore, neighbouring pixels might have a significant probability of having similar outputs: the lower the spatial frequency of the sampled image is, which in theory is good for RIP, the more pixels will have simultaneous pulses, which in practice is an added difficulty for a circuit that is designed to sum them asynchronously. In order to avoid overlapping pulses we must deliver them one by one and, for that reason, at the top of each column there is an event control unit (Ec.) designed to control the duration of each pulse and to prevent the pixels that share the same output bus from firing if another is already occupying it.

Due to the fact that the CS-CIS outputs are linear combinations of pixels values, it is impossible to obtain the characteristic of each pixel separately or to test the correct generation of each coefficient of the measurement matrix: we can only perform estimates of these elements from the circuit output which, a priori, is made of pseudo-random sums of their products. For this reason we added extra components and control signals that allow us to disable or bypass some parts of the CS-CIS in order to control its output linearly and perform specific sums from which we can then infer if all its components are working as intended.

4.1 DETAILED CHIP ARCHITECTURE

A prototype chip has been designed in a CMOS 0.18µm technology following the described methodology. The die size including pads is 3.17×2.23mm^2 (Fig. 27). It has 84 pads, of which one third is dedicated to power supply and ground connections. Here are some of the characteristics of the chip:

Technology	CMOS 0.18µm 1P6M
Die size (w. pads)	3174µm×2227µm
Pixel size	22µm×22µm
Fill factor	9.2%
Image Size	64×64
Photodiode type	n-well/p-substrate
Power supply	3.3V-1.8V
Nominal Frame rate	30fps
Clock Freq.	12.8-230.4MHz

Table 1: Summary of chip features.

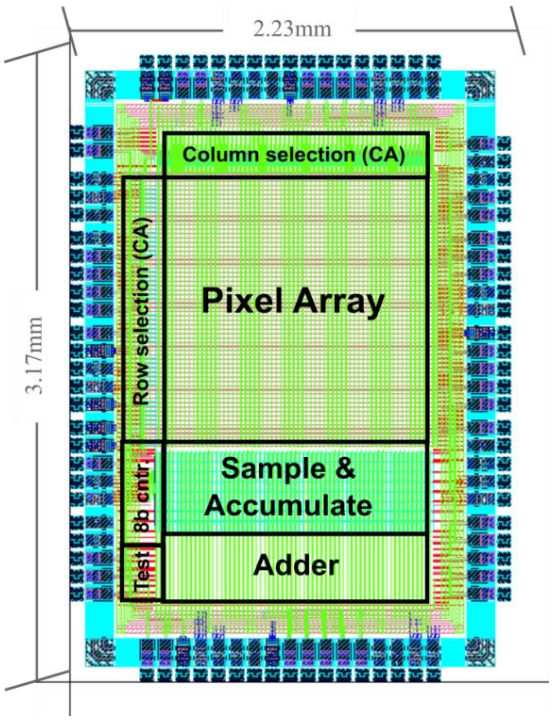

Fig. 27: Layout of the prototype sensor chip.

Furthermore, given the pixel's complex architecture, we are interested in studying its performance closely and, for that, we also introduced an extra pixel in a test area placed at the bottom left of the chip (Fig. 27), this pixel, which is an exact replica of those in the array but configured so that it is always "selected" (i.e. it is independent from the ECA), has independent analog and digital control signals and its output is directly accessible from an analog pad of the padring (so that we can also study the dynamic behaviour of its outgoing pulses and fine tune the analog control signals that eventually will be used to control the array). Note that in this area, besides the extra pixel, we also have included a copy of an event control unit (with the same structure of those that control the column busses in the array) to control the pixel pulse duration.

4.1.1 ELEMENTARY CELLULAR AUTOMATON

We were able to establish rules 30, 86, 135 and 149 as the ECA rules that better approximate random binary measurement matrices in terms of dynamic behaviour (see section 2.3) and output density (see section 2.5) and that would present a better performance in terms of oscillations and power consumption when used in a CMOS circuit (see sections 2.5 and 2.6). In the design of an ECA cell, beside the rule, we need to incorporate all the functionalities needed to test the CS-CIS as well as the registers needed to store the initial seed given as input to the ECA. In the end we opted to design a cell that could be used for rule 30 or 86 (see sections 2.4 and 2.6):

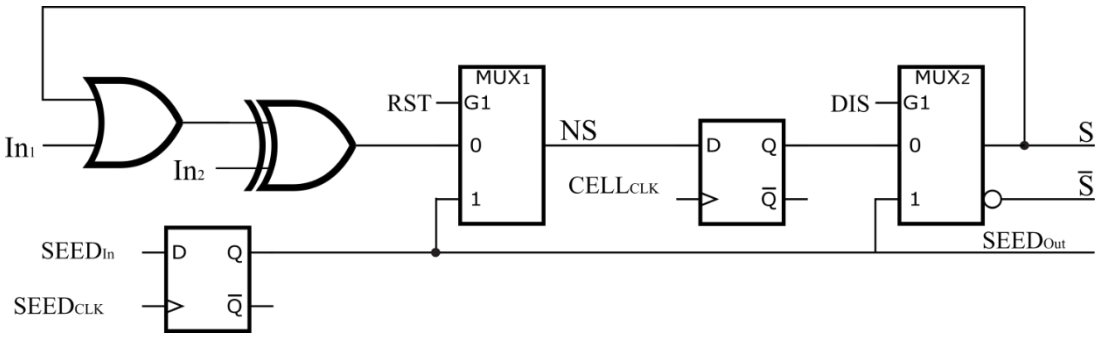

Fig. 28: Rule-30 ECA cell logic representation.

In Fig. 28 In_1 and In_2 inputs correspond to the outputs of the cell closest neighbours: if we connect to In1 the output of the right-hand neighbour and to In2 that of the left-hand neighbour than the rule implemented would be rule 30, otherwise, if the inputs were inverted, it would be rule 86. Being rule 30 (Table 2) the best known and most widespread [Wolf17], it became our choice.

The seed used to create the measurement matrix will be stored into the ECA using the bottom left static flip-flop of each cell. This flip-flop is connected to those of other cells through signals $SEED_{IN}$ and $SEED_{OUT}$. The seed is inserted into these flip-flops sequentially in what is known as

an initialization phase (see section 1.2), before the CS-CIS starts acquiring compressed samples. Clock signal $SEED_{CLK}$ will function only during this phase to input the seed and then it will be halted during normal operations so that the seed stored in these static flip-flops will not change over time, as long as the circuit remains properly powered, unless the ECA is loaded with a new seed.

In1	In2	S	NS
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	0

Table 2: Rule 30 truth table.

The next state in the evolution of rule 30 is implemented sequencing logic gates OR and XOR (see section 2.6). The current state is sent as input to multiplexor MUX_1, the other input of the multiplexor being the value of the seed flip-flop. When RST, the control signal of this two-input one-output multiplexor, is set to logic '0' the ECA will evolve normally. The frequency at which the cell will update its output is given by the frequency of clock signal $CELL_{CLK}$. By setting RST to logic '1' for a full clock cycle, it is possible to reset the temporal evolution of the ECA using the value stored in the seed flip-flop instead of the next state generated by rule 30. RST is common to all the cells. Clock signal $CELL_{CLK}$ will be used throughout the imager normal operations and it will set the frequency at which the measurement matrix is updated.

The only component left to describe in our cell is the two-input one-output multiplexor MUX_2. This multiplexor is essential for testing the pixel array. Since the evolution of a rule-30 ECA is pseudo-random, it would be difficult to use it to activate only the specific set of pixels that we want to test. When DIS, the control signal of MUX_2, is set to logic '1' the ECA are disabled and the seed flip-flops can be used as shift registers to directly influence pixel selection in order to activate a single column or row of the array.

4.1.2 PIXEL

Our approach to provide the number of bits prescribed by Eq. (11) is to time-encode the pixel values and employ time-to-digital conversion. The summation of pixels will then be realized in the digital domain, avoiding the requirement of a high SNR in the analog domain.

The architecture of a pixel in the array can be divided into five functions or functional elements (Fig. 29).

- Time-encoding (Photodiode and Comparator) is the sensing element in charge of transforming light intensity into electrical pulses;
- Trigger lock lets the generated event through and holds it in place until the whole circuit is reset;
- Pixel selection is connected to the outputs of the cellular automata, it is use to determine if the pulse will be part of a compressed sample or, if the pixel has not been selected, if the pulse will not reach the output bus;
- Event termination is an element that communicates with the event control unit in order to determine the duration of the outgoing pulse;
- Output control probes the bus and, if it is empty, releases the event.

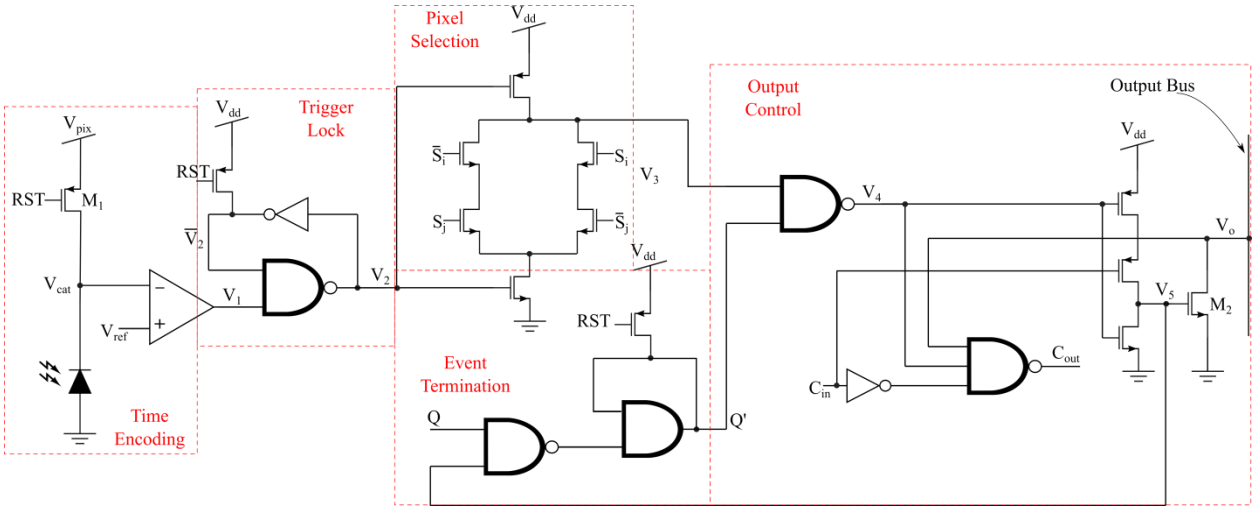

Fig. 29: Schematic of the elementary pixel.

Its Layout is presented in Fig. 30. In this section we will describe each of them in detail.

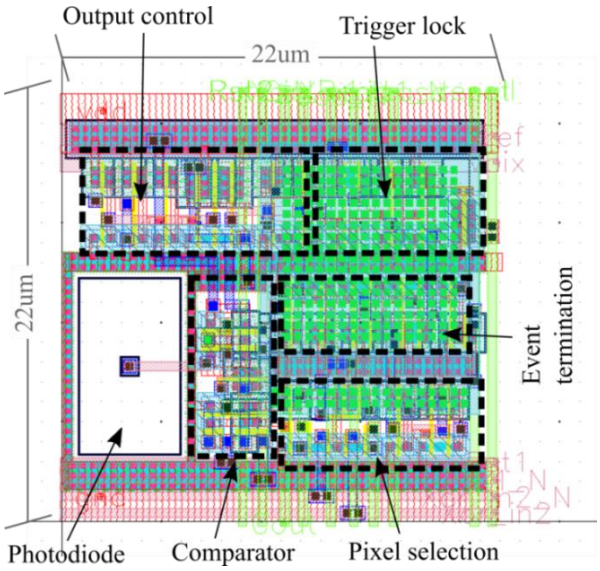

Fig. 30: Layout of the elementary pixel.

A time diagram of the signals shown in Fig. 29 is presented in Fig. 31, in this diagram we consider opposing S_i and S_j, hence an actively contributing pixel.

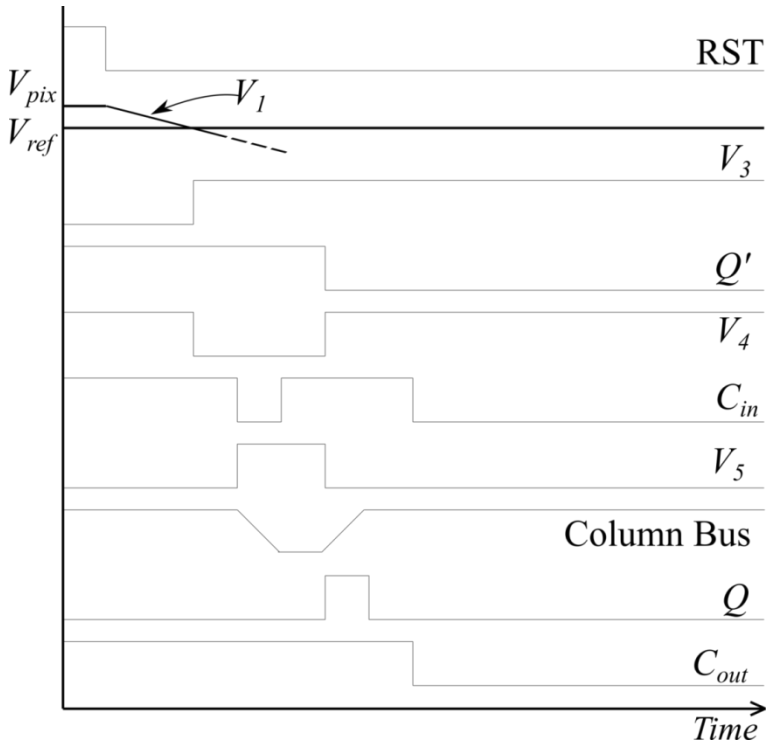

Fig. 31: Timing diagram of the pixel signals.

4.1.2.1 TIME-ENCODING OF LIGHT INTENSITY

The elementary pixel contains an integrating photodiode that discharges node V_{cat} at a rate determined by the photocurrent. This is depicted inside the 'Time-encoding of light intensity' box in Fig. 29. When V_{cat} crosses a reference voltage V_{ref}, (Fig. 31) a voltage comparator flips its output, V_1. It time-encodes the magnitude of the light intensity, what is described as pulse-modulation imaging [Chen11]. The pixel value is then contained in the period of time separating the reset of node V_{cat} and the moment in which V_1 turns from low to high. The lower (higher) the light intensity on the diode is, the longer (shorter) it takes to the comparator to switch. In this chip, both V_{pix} and V_{ref} can be adjusted on-line in order to adapt to different illumination conditions in real-time.

4.1.2.2 PROPAGATION OF THE ACTIVATION EDGE THROUGH THE TRIGGER LOCK

Signal V_1 is active in high, eliciting a rising edge in V_2 if this signal has not been activated before. If it has, the feedback of its own negative locks V_2 to logic '1' until the pixel is reset again. This block also prevents an instability that we will discuss further on.

4.1.2.3 PIXEL SELECTION

As previously mentioned, the contribution of each pixel to a particular compressed sample is determined by a combination of row and column selection signals, S_i and S_j, that are generated with the help of the ECA positioned around the sensor array (Fig. 26). These two signals are combined by an XOR gate implemented in wired-logic by 4 NMOS transistors (Fig. 29). The voltage V_3 is stuck at V_{dd} if selection signals S_i and S_j are equal. If not, V_3 is the inverse logic value of V_2. Using an XOR gate guarantees that the pixels contribute to the compressed sample in half of the possible combinations of S_i and S_j.

It is important to notice that this pixel selection unit is allocated right after the trigger lock because this helps reducing power consumption. If a pixel is not contributing to the compressed sample there is no reason to let the pixel activation front propagate, inducing changes in the subsequent nodes that are going to be discarded later.

4.1.2.4 EVENT TERMINATION CIRCUIT

Q is a global signal provided by a control unit present at the top of each column of the pixel array (Fig. 31). Let us consider that signal Q' is high. If Q' is in logic '1', then V_4 is the inverse of V_3, i. e. if the pixel is activated and is selected to contribute to the compressed sample, V_3 goes from logic '0' to '1' and V_4 goes from '1' to '0'. If signal C_{in} is low, this falling edge in V_4 induces a rising edge in V_5 which is the signal controlling driving transistor M_2. The column bus, whose voltage V_o is pulled up to V_{dd} by default, experiences a pull down driven by M_2. V_o will remain low if it was not for the event termination circuit.

The rising edge in V_5 is feedback to the event termination circuit, where it is inverted as long as Q is high. This causes Q' to fall to logic '0', switching back V_4 to logic '1' and then V_5 to logic '0', terminating the pulse that started before after a short delay.

The motivation to use the global pulse termination signal Q to establish the duration of the events instead of a local delay unit is to provide global control without introducing area and/or power consuming elements in the pixel. In particular, this unit probes the column bus and detects if it is being pulled down. Once the falling edge is detected, and after a user-controllable delay, Q rises enabling the termination of the pulse only in the pixel that has already turned M_2 on. This is verified by the NAND gate in the 'Event termination circuit' box (Fig. 29).

4.1.2.5 PIXEL OUTPUT CONTROL

As depicted in Fig. 29, all pixels in the same column of the array share the same column bus

to transmit its output pulse. As will be explained later, the time-encoding of the pixel value will be converted to digital by means of a time-to-digital converter, which in this case will be built with a clock and a counter. Of course, there is no a priori knowledge on the proximity of the values of the pixels and, therefore, how close in time will be the pulses emitted by the pixels. What is clear is that each one of them needs to be taken into account if we do not want to introduce additional errors in the image reconstruction from its compressed samples. In order not to skip any of the pulses, a token protocol is established so pixels that are being triggered close in time are only allowed to emit their pulse one after the other. This blocking mechanism needs to be parallel to all pixels so that the first pixel that delivers its event puts all other pixels on hold until its event is over. The release mechanism on the contrary has to be sequential so that, if there is more than one pixel in queue waiting to deliver its pulse, it will be impossible to have more than one of them active at the same time. In order to do so, each pixel receives a signal C_{in} from the pixel immediately above (Fig. 29), and sends a signal C_{out} to the pixel immediately below it. If there is no preceding pixel waiting to deliver a pulse through the column bus, C_{in} will be low. This enables the propagation of a falling edge in V_4 when it occurs into a rising edge in V_5. If C_{in} is high, however, this propagation is retained.

One pixel's C_{in} corresponds to its upper neighbour C_{out}. In order to be '0', three different conditions must hold, namely: its C_{in} is low, what means that there is no pixel above it that wants to deliver a pulse; V_4 is high, what means that either the pixel has not been activated or it has already delivered a pulse; and V_o is high, what means that the column bus is available. If any of these three conditions is not true than C_{out} will be stuck at the logic '1', thus preventing any of the pixels below it emitting a pulse through the column bus. A 3-input NAND gate is employed to combine the level at C_{in}, the pixel readiness to pull down the column bus and the feedback on the actual state of this column bus. This aggregated information is then sent as C_{out} to the pixels below. Using this logic each pixel will know that if V_o is set at V_{dd} and no pixel above is waiting to pull it down it is allowed to release its own event. Since V_o is fed back to this control block, when a pull down occurs, each pixel will simultaneously block the pixel immediately below through C_{out}. The blocking mechanism is parallel. On the contrary, when an event is over, its C_{out} turns to '0', so the pixels will be released sequentially in a top down fashion.

4.1.2.6 AUTO-ZEROING COMPARATOR

To design the comparator in Fig. 29, instead of using an operational amplifier (Op-Amp) we have used a digital inverter and, to reduce the influence of its offset, we have implemented an auto-zeroing scheme using a MiM capacitor on the top metal layers that does not show in Fig. 30

and three transmission gates (Fig. 32):

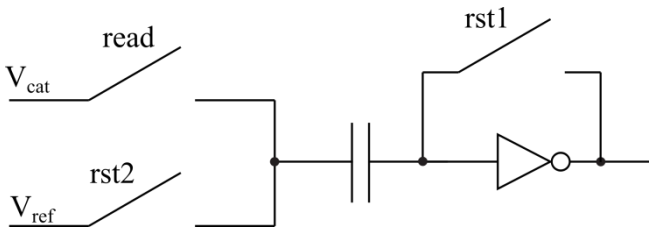

Fig. 32: Schematic of the autozeroing comparator.

The transmission gates are operated by three digital control signals (read, rst1 and rst2 of Fig. 33) that have not been mentioned in section 4.1.2.1:

Fig. 33: Timing diagram of the autozeroing comparator.

Enabling the transmission gate controlled by rst2, the MiM capacitor, placed before the inverter, is loaded with V_{ref}. While the capacitor is reaching the desired threshold, the transmission gate operated with rst1 is enabled as well to remove the inverter offset.

When M_1 (Fig. 29) is off, rst1 and rst2 are disabled and read is enabled. Thanks to the quick response of the digital inverter and the relatively short time needed for V_{cat} to cross V_{ref}, the resulting circuit operates as a comparator. It is important to time these control signals correctly, being RST the pixel rest signal of Fig. 29:

4.1.2.7 CONSIDERATIONS ON SIGNAL STABILITY OF THE ADOPTED SOLUTIONS

The components above described were initially designed and simulated separately but, even though they all worked as intended, when placed together to run a joint simulation the resulting behaviour was inconsistent. The three main causes of this behaviour were:

- The autozeroing comparator slow changing input.
- The closed loop created by the output control unit and the event termination.
- A cross-pixel oscillatory behaviour in the closed loop created by two output control units sequentially connected by the control cascade C_{in}-C_{out} and sharing the same bus.

A consequence of the fact that V_{cat} changes slowly with respect to the response of the technology is that V_1 remains undetermined for as long as the voltage at the cathode stays close to the reference voltage (Fig. 31). As such, if the pixel selection circuit were to follow the comparator directly, its PMOS and NMOS transistors would spend a relatively long time switching on and off simultaneously due to the undetermined input. This is how we first attempted to design the pixel but, in these conditions, when the circuit was simulated along with parasitic components generated by the post layout extraction, we observed an oscillatory behaviour in $V_{3,}$ which kept bouncing back and forth from V_{dd} to GND and did not stop until V_{cat} reached a voltage well below V_{ref} or, in other words, until V_1 became once again determined approaching V_{dd}. These oscillations were so fast that the event termination circuit could not respond in time and the event termination circuit did not have time to raise Q. These oscillations were transferred all the way to the output bus and to the whole column bus thus invalidating the whole design. To stop them it was necessary to include the trigger lock presented in Fig. 29. Its feedback loop is introduced in order to present a stable V_2 as input to the pixel selection circuit independently of the state of V_1: as soon as V_1 leaves logic '0' V_2 switches and remains stable thanks to its own feedback mechanism.

The second source of instability was due to the fact that, when the event termination NAND received Q and the output control unit had M_2 turned on, the resulting configuration started oscillating between V_{dd} and GND. To make the event termination work properly it was necessary to break this loop by adding along the feedback line a second trigger lock. This solution allowed the event termination to work as expected with the added benefit that, since all pixels share Q, once an event was terminated and the AND gate switched its output, it would remain stable until the end of acquisition avoiding possible repetitions of events from the same pixel.

The cross-pixel oscillations were introduced in a similar fashion when V_5 was delivered by a NOR instead of the gated inverter seen in Fig. 29. When an event occurred, the output of the NOR gate acted over M_2. The change in state that this produced on the Output Bus was then indirectly looped into one of the NOR gate inputs through signal C_{in} arriving from the pixels above. It was not possible to remove this instability devising a solution similar to the one used in two previous cases because, on the one hand, if we were to lock the NOR gate output, we would have been sending an endless event and, on the other hand, since the sequence of C_{in}-C_{out} is common to all the pixels, blocking it everywhere else but at the NOR gate level would have prevented this sequence of control signals from working properly. The problem was resolved by removing the pull down transistor of the NOR input that received C_{in}. In doing so, even if the information transmitted by the chain of C_{in}-C_{out} locks all pixels, the one that is already sending

its output will continue to do so until Q is sent. The resulting circuit takes the form of the gated inverter seen in Fig. 29.

4.1.3 EVENT CONTROL UNIT

Q (Fig. 29) is a global signal provided to all the pixels of a column by the event control unit present at the top of the column itself. It is flipped from logic '0' to logic '1' when this unit detects a pull down action on Output Bus .i.e. when a pixel of the column has M_2 turned on. How much time passes between the detection of a change in Output Bus and the flipping of Q depends upon the value of an external analog control signal, V_{str} (Fig. 34):

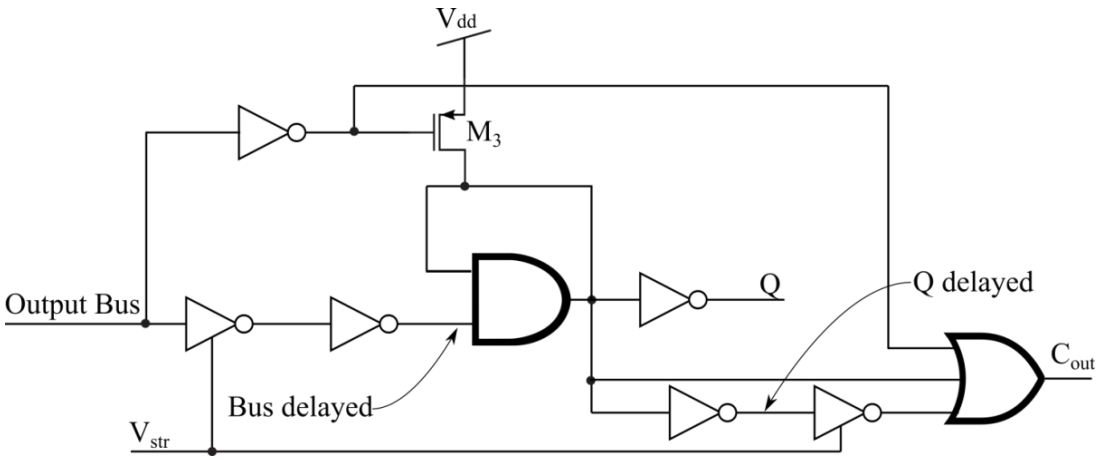

Fig. 34: Schematic of the event control unit.

By design, the state of Q must directly depend upon the state of Output Bus but, if the response of this unit were to occur as soon as a pull down action is detected, we would face two problems, the first is that we would not have control on the duration of an event, which would be immediately quenched by this system, and the second is that we would incur in the same oscillatory behaviour described in section 4.1.2.7, which affects the output controls of two consecutive pixels. To solve both these problems, an external analog control signal V_{str} is used to regulate the starvation level in a couple of inverters that in turn are used to regulate a delay on a sequence of logic gates that connects Output Bus to Q.

Controlling the duration of the events sent by the pixels is not the only function that this unit has. Since the first pixel of each column lacks an upper neighbour from which to receive control signal C_{in} (Fig. 29), this unit must provide it.

As Fig. 34 shows that the state of Output Bus and the external signal V_{str} are the only inputs of event control unit so Fig. 35 shows the timing diagram of the signals within this system:

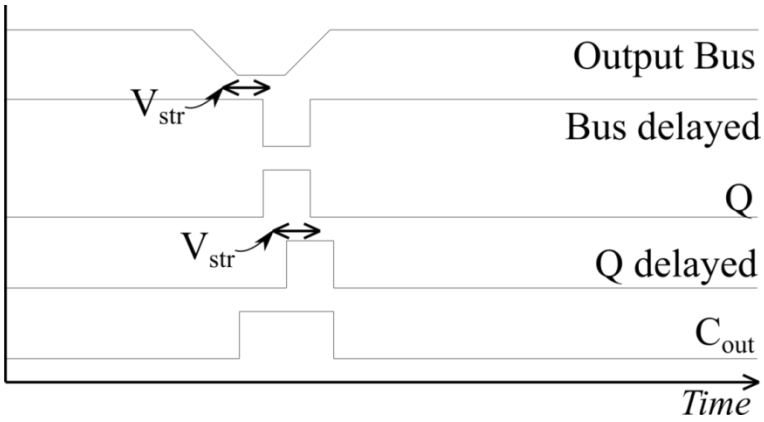

Fig. 35: Timing diagram of the Event Control Unit Signals.

When an event generated by a pixel changes the state of Output Bus to logic '0', using a starved inverter to slow down its propagation, this logic '0' is transmitted to an AND gate. This gate and has its output directly fed back to the other one of its inputs and is used to hold signal Q at logic '1' until the Bus returns to V_{dd}. When it happens, i.e. an event has ended, transistor M_3 resets the AND gate returning Q to logic '0'.

The output of the AND is further divided into two more lines, one of them being delayed using another starved inverter with the same V_{str}. A three-Input OR gate is used to combine these lines with the inverse of Output Bus providing the control signal C_{out} to the first pixel of the column. This second starved inverter is present so that, even if an event ends, the pixels will stay locked for a time proportional to that of the event itself. This precaution is set in place to avoid two consecutive events to occur too close to one another so that the readout system cannot discern them.

4.1.4 8-BIT COUNTER

As shown in the section 3.2.2, using PWM, each pixel sends only one pulse per compressed sample. The burden of encoding the pixels outputs using an appropriate amount of bits is left to the S&A and the counter. In PWM, the relationship between photo-generated current I_{ph} and integration time t is hyperbolic:

$$I_{ph}t = C_{ph}(V_{pix} - V_{ref}) \tag{48}$$

being C_{ph} the photodiode capacitance, V_{pix} the pixel reset voltage and V_{ref} the reference voltage at the autozeroing comparator positive input (Fig. 30). As such, PWM needs a counter that counts at a decreasing frequency from 255 backward to 0. For UMC180nm, through simulations, we estimated the photodiode capacitance to be $C_{ph} = 60\text{fF}$. Given $V_{pix} - V_{ref} = 0.1\text{V}$, and a

range of currents between 300pA and 30pA it is possible to compute the time needed to collect an event, which varies between 2μs and 20μs, in line with the requirement of Eq. (47).

The clock of the counter will need to switch 256 times starting with an offset of 2μs and finishing its operation 18μs later. From these values we can compute the range of frequencies at which the counter clock needs to operate:

$$\begin{cases} f_{\text{int}} = 230.4\text{MHz} \\ f_{\text{end}} = 12.8M\text{Hz} \end{cases} \tag{49}$$

being f_{int} the initial frequency needed to count 256 times if the pulses where to arrive all just after the offset (i.e. highly illuminate image) and f_{end} the final frequency needed to count 256 times if the pulses where to arrive at the end of the integration period (i.e. scarcely illuminated image). This varying frequency is a direct result of the asynchronous pulse width modulation architecture presented in section 3.2.2 in which the counter needed to be fitted with a linearly changing input frequency in order to obtain a direct proportionality between the photo-generated current and the timing of an event. This value of f_{int} leaves the duration of a single pulse to 5ns which is still feasible in CMOS 0.18μm 1P6M technology.

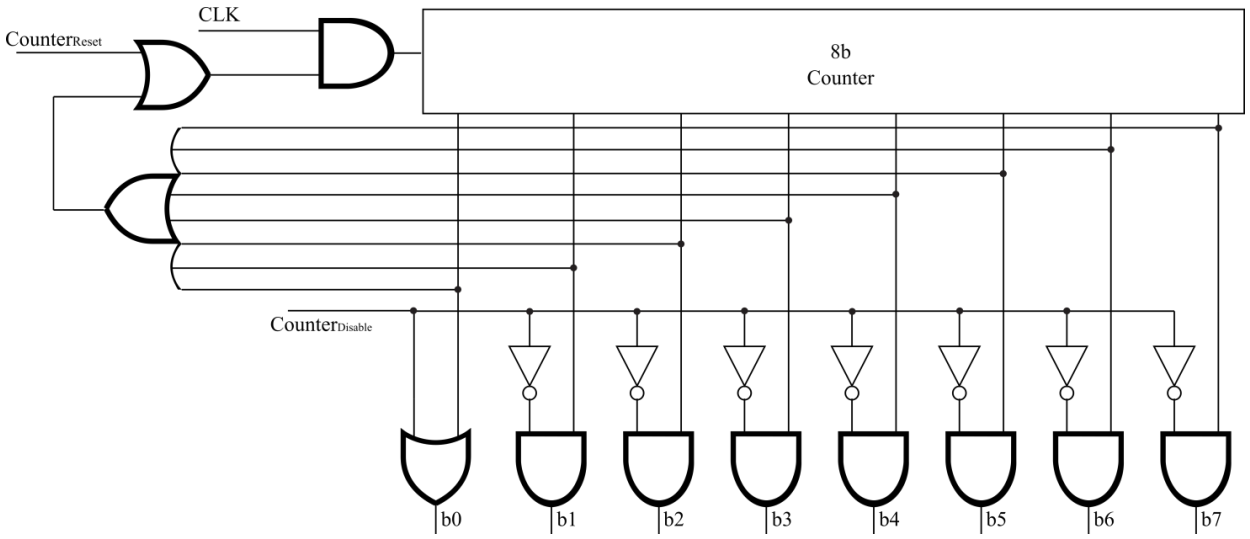

Fig. 36: Schematic of the 8-bit Counter.

Fig. 36 represents the floorplan of the 8-bit counter. The first thing that we notice is the eight-input OR gate. This element is commonly used in counters as flag to alert of possible overflows i.e. when the word in a countdown has reached ‘00000000’ and the next clock cycle would reset it to ‘11111111’. It receives as inputs all of the bits of the counter and its output becomes logic ‘0’ only when the countdown has reached the end. In our circuit we use this piece of information as a safeguard to prevent the counter from resetting even when clock signal CLK is left running.

At the end of each compressed sample acquisition, to reset the counter, signal $\text{Counter}_{\text{Reset}}$ is set at logic '1' for a short duration (Fig. 37):

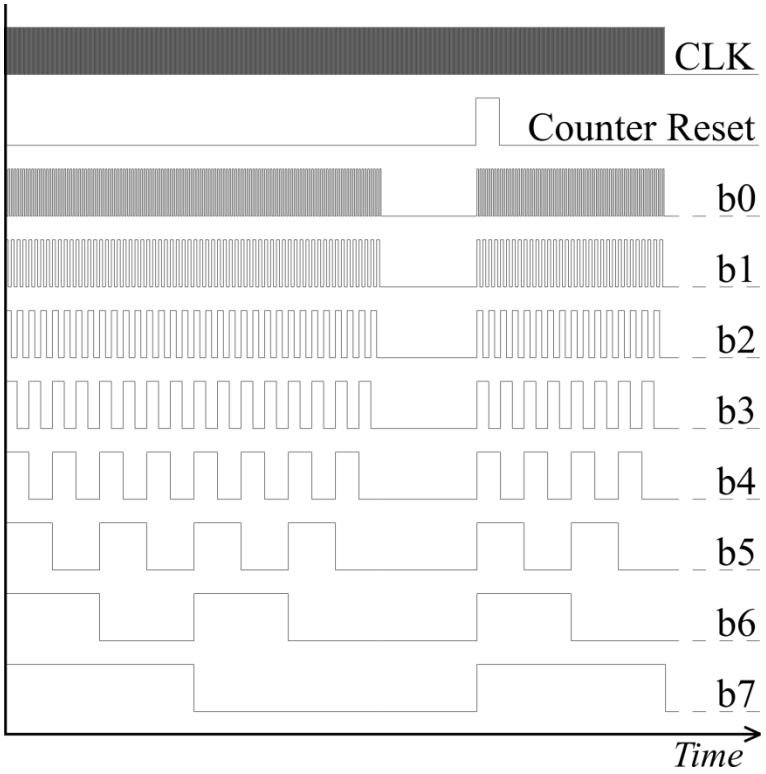

Fig. 37: Timing diagram of the 8-bit Counter floorplan.

To generate a single control signal with the varying frequency needed to properly operate the counter might not be simple. A first order approximation and a viable alternative to a varying signal would be to generate a sequence of signals of constant frequencies, from f_{int} to f_{end}, and feed them to the counter's input through a MUX by switching them at regular intervals from the fastest to the slowest. In doing so we would simplify the design of the controller but we might incur in small approximation errors or tension drops during switches that might lead us to misjudge the number of cycles of the counter. This is another reason behind the implementation of this reset function: it is a safety measure to avoid miscalculations in the number of cycles that might occur if we used suboptimal control signals. If we used a more complex and reliable control signal it would be sufficient to set $\text{Count}_{\text{Reset}}$ to logic '1' permanently to disable this additional safeguard.

We do not have direct means to access the pixels outputs. Their correct behaviour can only be inferred from the CS-CIS output. The matrix generated by the ECA can only be accessed indirectly through the same channels. This is also true for the 8-bit counter when we factor the asynchronicity of the pixels responses needed to propagate its counts and the fact that these counts are only registered within the S&A elements. It is essential that we devise a way to

separate the contributions of all this components so that we can use the same outputs to infer their behaviour. On the one hand we will need to find a way to disable the counter to facilitate possible tests of pixel array and ECA and on the other hand we need to find a way to propagate the counts of the counter without depending upon the pixels events.

The logic gates placed at the exit of each bit in Fig. 36 serve as solution to the first problem indicated above as they can be used to disable the counter. By setting digital signal $Counter_{Disable}$ to logic '1' it is possible to replace the counter output with the fixed word '00000001'. In doing so, each pixel pulse will be recorded as a unit value within the S&A elements.

We could then disable an ECA (Fig. 28) and use the seed registers to activate only pixels in known positions. For instance, once a single row or column of the array is active, if all the pixels are working properly, the output of the CI-CIS in base-ten will be 64. By activating the rows and columns one after the other sequentially it is possible to draw a map of which pixels are working properly and which are not. In the same way, once we tested that the pixels are working, we could let the ECA evolve naturally and the sum of the non-zero elements of a row of the measurement matrix should correspond to the output of CS-CIS.

We will describe how to remove the dependency of the propagation of the counter outputs from the pixel responses in the next section. Fig. 37 shows the timing diagram of the 8-bit counter during normal operations.

4.1.5 SAMPLE & ACCUMULATE

As we expressed in Chapter 3 we want to realise the summation of pixels contributions in the digital domain to avoid the requirement of a wide dynamic range in the analog domain. To do so we have pulse-width modulated the pixels' outputs in order to time-encode their values into pulses as expressed in section 4.1.2 and now we need associate the timing of these events with the countdown generated by the counter presented in section 4.1.4 in order to perform their time-to-digital conversion. This association is carried out by the S&A circuit. The architecture of 1 bit of the S&A can be divided into three functions or functional elements (Fig. 38):

- Full Adder, the element at core of the S&A functionality, it is used to create the sum of the pixel contributions;
- Input handling is a logic component that, depending on the mode of operation, connects the S&A input to the column bus for acquisition or to other S&As to add the contributions of all columns in order to generate a compressed sample;
- Sum Storage is a 1-bit memory that is used to store the result of the full adder after each step in a close loop so that the system can actually accumulate the various contributions.

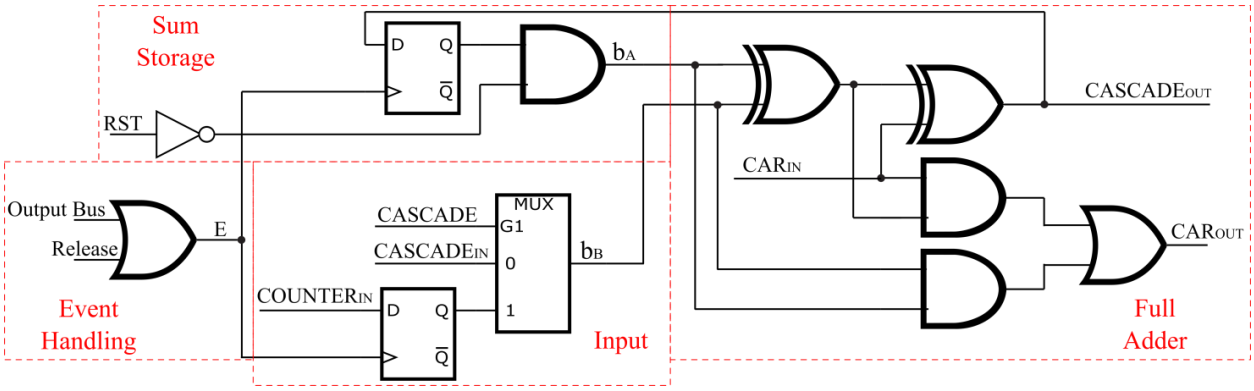

Fig. 38: Schematic of 1 bit of the sample & accumulate.

In Fig. 38 we also added a fourth component, the Event handling, that is not part of a single bit but that is common to all bits, nonetheless, since it is essential to understand how the events are transmitted to the S&A, we have reported it here.

4.1.5.1 FULL ADDER

To get an idea of the structure of the overall S&A, all bits in a S&A element/column are disposed vertically from the least-significant bit on top and the most significant bit below. Each bit has to its right and to its left the bits that occupy the same positions in the S&A elements of the adjacent columns.

On the right side of Fig. 38 we have a full adder that receives as inputs CAR_{IN}, the bit carried in from the lesser-significant stage above, and the outputs b_A and b_B of the Sum Storage and Input circuits. This full adder generates the output carry, CAR_{OUT}, which goes to the more-significant stage below and the sum of b_A, b_B and CAR_{IN}, here named $CASCADE_{OUT}$. $CASCADE_{OUT}$ is divided into two lines, the first is fed back to Sum Storage and the second is transmitted to the bit in the column to its right. The column to its right will receive this value as $CASCADE_{IN}$. During integration, RST, and Release are set to logic ‘0’ and CASCADE is set at logic ‘1’, as such, given the configuration of the MUX, $CASCADE_{OUT}$ is effectively only used as input for the flip-flop in Sum Storage.

4.1.5.2 INPUT

The flip-flop receives as input $COUNTER_{IN}$ which is one of the bits of the 8-bit counter. The clock signal of the flip-flop is E and it is handled by the Event Handling Unit, which remember it is located outside of the bit and is common to all bits in a S&A column.

Since CASCADE is set at logic ‘1’, at each double-flip of E, the corresponding counter output, (see Fig. 36), is moved to b_B and summed to the bit present in Sum Storage. When the

acquisition period is over, the global signal CASCADE is set to logic ‘0’, the flip-flop is then ignored (and the counter disconnected) and b_B takes the value of the bit at the same stage in the column to its left as described in the section above. This generates an avalanche effect that start adding the contribution of the left-side neighbours to that of the right-side neighbours. After some delay due to propagation, the rightmost S&A will show, at its output, the sum of all the S&A elements connected sequentially before it.

4.1.5.3 SUM STORAGE

This circuit is responsible for storing the actual value of the current sum, $CASCADE_{OUT}$, in its flip-flop, so that, when E double-flips another time, this value can be added to the new value of $COUNTR_{IN}$ and, by doing so we keep on adding new counter values on top of the already existing sum.

After the compressed sample has been recorded off-chip, RST is set to logic ‘1’ and an extra flip of E effectively empties the all the sums stored in all the flip-flops of all the bits and in doing so resets the S&A elements readying them for the collection of the next compressed sample.

4.1.5.4 EVENT HANDLING

This component serves a double purpose; firstly it directly takes the output bus of a column in the pixel array and transforms it into signal E effectively creating the connection that we were missing in order to associate pixel events to counter values and generate compressed samples. But, this unit can also be used to create artificial pulses by means of the global signal Release (Fig. 38). In doing so the counter and S&A can be tested generating artificially controlled events from the outside without depending upon the pixel array and ECA effectively resolving the second problem presented in section 0. This control signal is also part of the readout and reset phases of the S&A and it does so by flipping E twice, the first time to prepare the column sum before CASCADE is set to logic ‘1’ and the second time, in conjunction with RST to reset the S&A and restart the acquisition phase with empty registers, as shown in Fig. 39.

4.1.5.5 SIMPLIFICATIONS

It has to be noted that, the first and last bits of the 20 bits S&A have been simplified since there is no need to implement CAR_{IN} and CAR_{OUT} respectively. Furthermore, from bits 9 to bit 20, $COUNTER_{IN}$ and its corresponding flip-flop are replaced by a connection to ground because the counter only has 8 bits. the multiplexors have been replaced with an AND gate that receives CASCADE as one of the input and $CASCADE_{IN}$ as the other. This solution greatly decreases the

area occupation of these elements making the final design more compact.

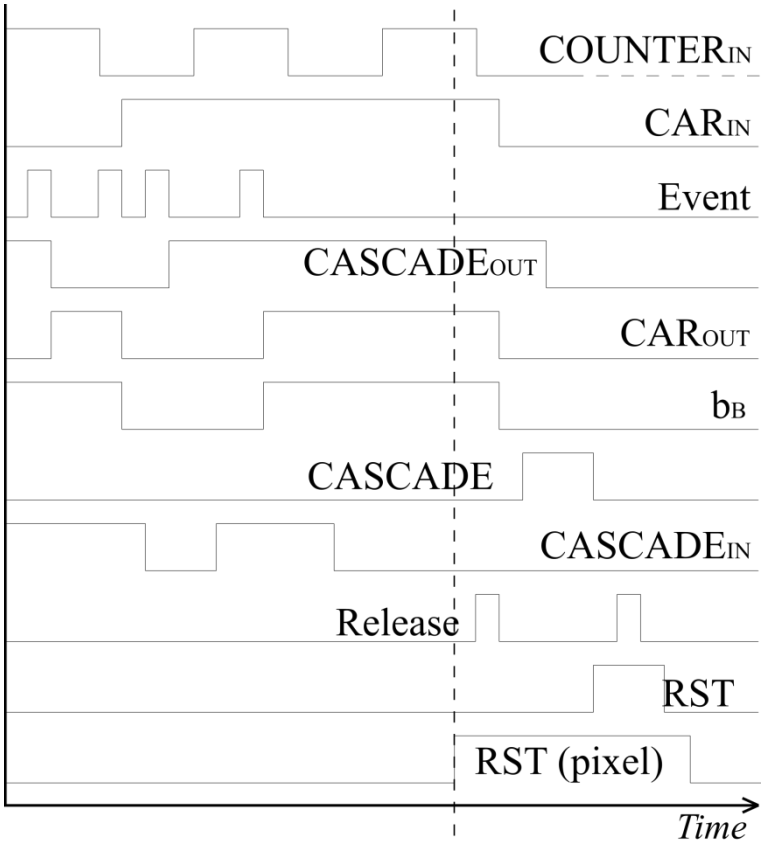

Fig. 39: Timing diagram of 1 bit of the sample & accumulate during acquisition (right) and compressed sample generation (left).

Fig. 39 shows the timing diagram of the signals within 1 bit of a S&A element during a typical operation. The rising edge of RST (Fig. 30) determines the end of the acquisition period and the beginning of the phase in which the S&A of the various columns are connected among each other to deliver the compressed sample. Its falling edge establishes the start of a new integration period.

4.1.6 TEST COMPONENTS: EXTRA PIXEL AND EVENT CONTROL UNIT

Of all of the control signals that must be coordinated and tuned to operate the CS-CIS, three of them are analog and their values are difficult to set properly because they may vary depending on light conditions and the gain of the amplifiers used to control them depends on the technology so a fine adjustment is necessary but we do not have direct access to the pixels outputs, i.e. we cannot measure their direct impact on the chip overall performance. These signals are V_{pix} V_{ref} and V_{str} respectively. To resolve this problem we have decided to introduce an extra pixel and event control unit in a corner of the pixel. On the PCB used for testing, three potentiometers can be used to set these variables that are connected to three floating pins. It is possible to connect

these pins to Vpix Vref and Vstr of the test pixel whose output is directly accessible from an analog pad of the padring. This makes it possible to tune these signal values to obtain the type of pulse desired. Once that is achieved, these signals can be transmitted to the array instead of to the test pixel. For the time being this adjustment will be manual, however, it would possible to to exploit this extra pixel design an automated adjustment system that would favour the sensor adaptability to different light conditions.

4.2 PROTOTYPE PACKAGE AND PINAGE

A prototype chip named ‘CSImager1’ has been designed in a CMOS 0.18µm technology. The die size including pads is 3.17×2.23mm^2. The chip has been packaged in a CD-PGA84 ceramic capsule (Fig. 40).

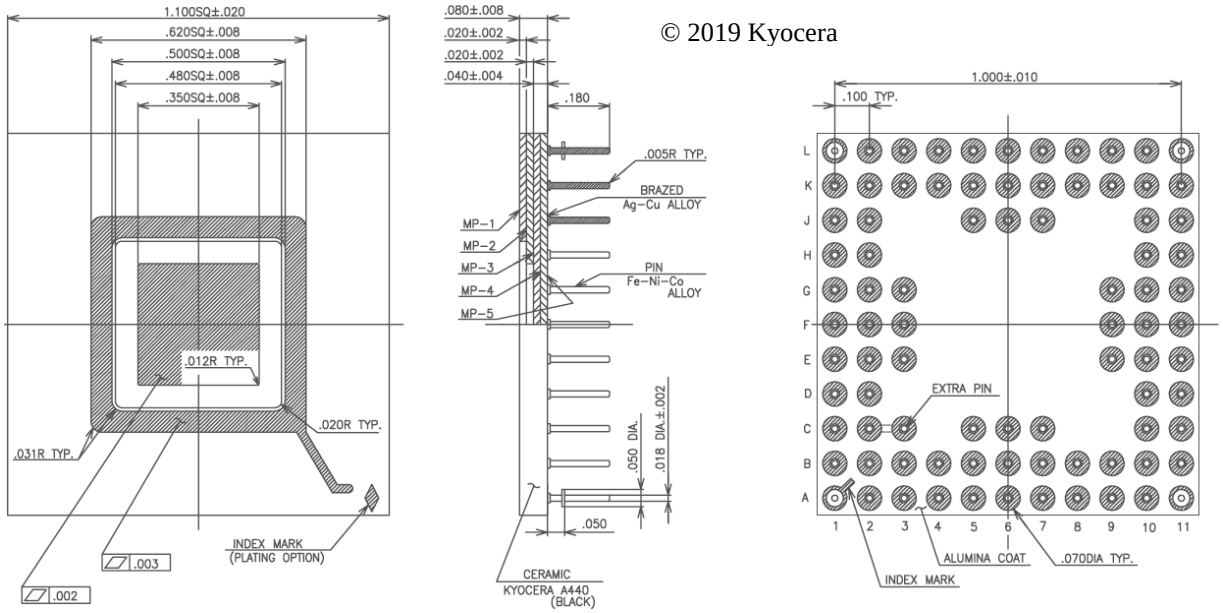

Fig. 40: CD-PGA84 ceramic capsule [Kyoc19] © 2019 Kyocera.

Its connection map is presented in Fig. 41:

WIRE BOND PAD / CONNECTOR PIN INTERCONNECTION PLAN

W/B NO.	PIN NO.
1	B 2
2	C 2
3	B 1
4	C 1
5	D 2
6	D 1
7	E 3
8	E 2
9	E 1
10	F 2
11	F 3
12	G 3
13	G 1
14	G 2
15	F 1
16	H 1
17	H 2
18	J 1
19	K 1
20	J 2

W/B NO.	PIN NO.
21	L 1
22	K 2
23	K 3
24	L 2
25	L 3
26	K 4
27	L 4
28	J 5
29	K 5
30	L 5
31	K 6
32	J 6
33	J 7
34	L 7
35	K 7
36	L 6
37	L 8
38	K 8
39	L 9
40	L10

W/B NO.	PIN NO.
41	K 9
42	L11
43	K10
44	J10
45	K11
46	J11
47	H10
48	H11
49	F10
50	G10
51	G11
52	G 9
53	F 9
54	F11
55	E11
56	E10
57	E 9
58	D11
59	D10
60	C11

W/B NO.	PIN NO.
61	B11
62	C10
63	A11
64	B10
65	B 9
66	A10
67	A 9
68	B 8
69	A 8
70	B 6
71	B 7
72	A 7
73	C 7
74	C 6
75	A 6
76	A 5
77	B 5
78	C 5
79	A 4
80	B 4

W/B NO.	PIN NO.
81	A 3
82	A 2
83	B 3
84	A 1

S/R	N C
D/A	N C
EXTRA PIN (C3)	C 2

Fig. 41: CD-PGA84 connection map [Kyoc19] © 2019 Kyocera.

4.2.1 BONDING DIAGRAM

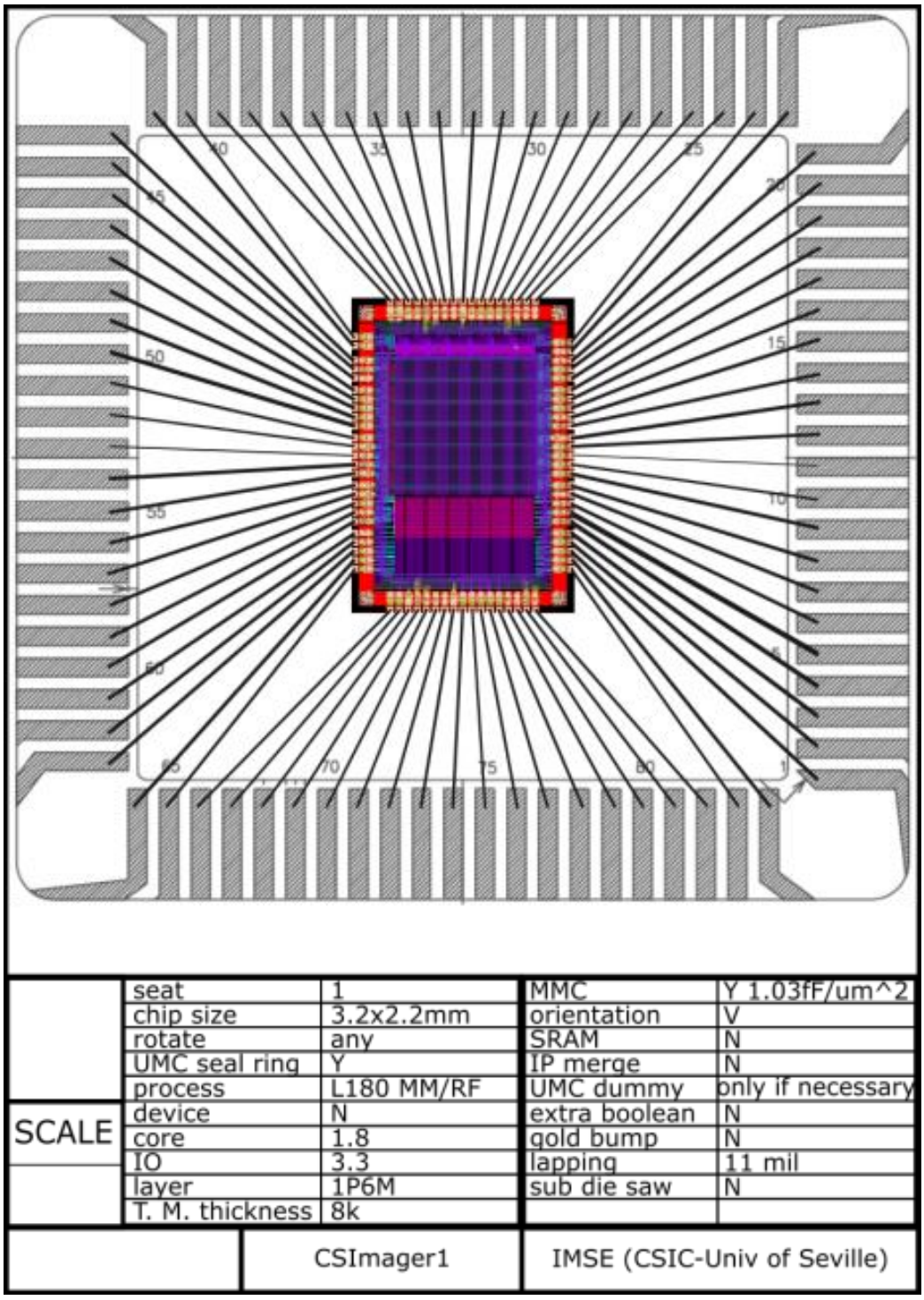

Fig. 42: Bonding Diagram of CS-CIS 'CSImager1'.

4.2.2 PINAGE OVERVIEW

Seven different types of pads have been employed in the design of this chip, including power and ground pads, analog and digital inputs and outputs. Table 3 contains a short description of

the pads. Given the layout of the connections of Fig. 42, the pin map bottom and top views are presented in Table 4 and Table 5 respectively:

Cell name	Type	Description	N.	Volt. level	Pulled		Buff	ESD
					up	down		
VCC3IOD_DigIO	VDD	Power supply	8	3.3				x
VCCKD_DigCore	VDD	Power supply	9	1.8				x
GND3IOD_DigIO	GND	Ground connection	8	0				x
GNDKD_DigCore	GND	Ground connection	9	0				x
XMD_DigI	DI_IN	Digital Input	19	{0,3.3}			x	x
YA2GSD_DigO	DI_OUT	Digital Output	21	{0,3.3}			x	x
ANA_ESD	AN_IN AN_OUT	Analog I/O	8	[0-1.8]				x

Table 3: Pad types used in the design.

	1	2	3	4	5	6	7	8	9	10	11	
L	VCCO	GND	VCCO	GND	READ	GND	VCCO	CEDI	VSTR	VREF	SECL	L
K	S1	GND	SEEX	VCCK	RST2	RESE	VCCK	CERS	SEED	VCCK	GND	K
J	S2	S0			RST1	VPIX	GND			GND	VCCO	J
H	S4	S3								CECL	CODI	H
G	VPIX	GND	S5						VPIX	CORE	REDY	G
F	VCCK	S7	S6						VCCK	COCL	GND	F
E	S8	S9	GND						VRPT	VPPT	VSPT	E
D	VCCO	VCCK								VCCK	RSPT	D
C	GND	S11	NC		S15	S19	VPIX			VCCO	GND	C
B	S10	S12	S14	GND	S16	VCCK	VCCO	PUPT	REPT	CASC	GND	B
A	S13	VCCK	GND	VCCO	S17	S18	GND	GND	R2PT	R1PT	RESA	A
	1	2	3	4	5	6	7	8	9	10	11	

Table 4: Pin map of the chip (bottom view).

	11	10	9	8	7	6	5	4	3	2	1	
L	SECL	VREF	VSTR	CEDI	VCCO	GND	READ	GND	VCCO	GND	VCCO	L
K	GND	VCCK	SEED	CERS	VCCK	RESE	RST2	VCCK	SEEX	GND	S1	K
J	VCCO	GND			GND	VPIX	RST1			S0	S2	J
H	CODI	CECL								S3	S4	H
G	REDY	CORE	VPIX						S5	GND	VPIX	G
F	GND	COCL	VCCK						S6	S7	VCCK	F
E	VSPT	VPPT	VRPT						GND	S9	S8	E
D	RSPT	VCCK								VCCK	VCCO	D
C	GND	VCCO			VPIX	S19	S15		NC	S11	GND	C
B	GND	CASC	REPT	PUPT	VCCO	VCCK	S16	GND	S14	S12	S10	B
A	RESA	R1PT	R2PT	GND	GND	S18	S17	VCCO	GND	VCCK	S13	A
	11	10	9	8	7	6	5	4	3	2	1	

Table 5: Pin map of the chip (top view).

4.2.3 Signals Overview

Table 6 and Table 7 describe the pin assignment complemented with a short description of the purpose of the corresponding signals. This table has 5 columns:

- W/B. NO.: Is the number of the finger that the wire bond is attached to (Fig. 42).
- PIN NO.: Displays the coordinates of its corresponding pin (Fig. 41).
- Short Name: Name of the pins as included in Table 4 and Table 5.
- PAD Name: Indicates the name of the pad as included in the Layout design in Cadence.
- Description: Brief definition of the signal purpose according the circuit description given in the section above.

W/B NO.	PIN NO.	Short Name	PAD Name	Description
1	B_2	S12	S12	13th bit of the last S&A (CASCADEOUT of Fig. 38)
2	C_2	S11	S11	12th bit of the last S&A (CASCADEOUT of Fig. 38)
3	B_1	S10	S10	11th bit of the last S&A (CASCADEOUT of Fig. 38)
4	C_1	GND	GNDK	Ground
5	D_2	VCCK	VCCK	1.8V constant voltage supply for the chip core
6	D_1	VCCO	VCC30	3.3V constant voltage supply for the chip padring
7	E_3	GND	GNDO	Ground
8	E_2	S9	S9	10th bit of the last S&A (CASCADEOUT of Fig. 38)
9	E_1	S8	S8	9th bit of the last S&A (CASCADEOUT of Fig. 38)
10	F_2	S7	S7	8th bit of the last S&A (CASCADEOUT of Fig. 38)
11	F_3	S6	S6	7th bit of the last S&A (CASCADEOUT of Fig. 38)
12	G_3	S5	S5	6th bit of the last S&A (CASCADEOUT of Fig. 38)
13	G_1	VPIX	VPIX	Reset voltage of the pixel array photodiodes (Fig. 30)
14	G_2	GND	GNDK	Ground
15	F_1	VCCK	VCCK	1.8V constant voltage supply for the chip core
16	H_1	S4	S4	5th bit of the last S&A (CASCADEOUT of Fig. 38)
17	H_2	S3	S3	4th bit of the last S&A (CASCADEOUT of Fig. 38)
18	J_1	S2	S2	3rd bit of the last S&A (CASCADEOUT of Fig. 38)
19	K_1	S1	S1	2nd bit of the last S&A (CASCADEOUT of Fig. 38)
20	J_2	S0	S0	1st bit of the last S&A (CASCADEOUT of Fig. 38)
21	L_1	VCCO	VCC30	3.3V constant voltage supply for the chip padring
22	K_2	GND	GNDO	Ground
23	K_3	SEEX	SEED_EXIT	Binary seed output of the ECA
24	L_2	GND	GNDO	Ground
25	L_3	VCCO	VCC30	3.3V constant voltage supply for the chip padring
26	K_4	VCCK	VCCK	1.8V constant voltage supply for the chip core
27	L_4	GND	GNDK	Ground
28	J_5	RST1	RST1	1st reset signal for the array comparators (Fig. 32)
29	K_5	RST2	RST2	2nd reset signal for the array comparators (Fig. 32)
30	L_5	READ	READ	Read signal for the array comparators (Fig. 32)
31	K_6	RESE	RESET	reset signal for the pixel array(Fig. 30)
32	J_6	VPIX	VPIX	Reset voltage of the pixel array photodiodes (Fig. 30)
33	J_7	GND	GNDO	Ground
34	L_7	VCCO	VCC30	3.3V constant voltage supply for the chip padring
35	K_7	VCCK	VCCK	1.8V constant voltage supply for the chip core
36	L_6	GND	GNDK	Ground
37	L_8	CEDI	CELL_DISABLE	Disable signal for the ECA (Fig. 28)
38	K_8	CERS	CELL_RST	Reset signal for the ECA (Fig. 28)
39	L_9	VSTR	IREF	Control voltage for the starved inverters (Fig. 34)
40	L_10	VREF	VREF	Voltage reference for the pixel array (Fig. 30)
41	K_9	SEED	SEED	Binary seed input for the ECA (Fig. 28)
42	L_11	SECL	SEED_CLK	Clock to drive the seed into position (Fig. 28)
43	K_10	VCCK	VCCK	1.8V constant voltage supply for the chip core
44	J_10	GND	GNDK	Ground
45	K_11	GND	GNDO	Ground
46	J_11	VCCO	VCC30	3.3V constant voltage supply for the chip padring
47	H_10	CECL	CELL_CLK	Clock signal for the ECA (Fig. 28)
48	H_11	CODI	COUNT_DISABLE	Signal to disable the counter (Fig. 36)
49	F_10	COCL	COUNT_CLK	Clock signal for the time to digital conversion
50	G_10	CORE	COUNT_RESET	Signal to restart the counter (Fig. 36)

Table 6: Pin assignment (first half).

W/B NO.	PIN NO.	Short Name	PAD Name	Description
51	G_11	REDY	READY	Signal to prepare the S&A output (Fig. 38)
52	G_9	VPIX	VPIX	Reset voltage of the pixel array photodiodes (Fig. 30)
53	F_9	VCCK	VCCK	1.8V constant voltage supply for the chip core
54	F_11	GND	GNDK	Ground
55	E_11	VSPT	IREF_Pixel_Test	Control voltage for the starved inverter of the test pixel
56	E_10	VPPT	VPIX_Pixel_Test	Reset voltage of the test pixel photodiode
57	E_9	VRPT	VREF_Pixel_Test	Voltage reference for the test pixel comparator
58	D_11	RSPT	RESET_Pixel_Test	reset signal for the test pixel
59	D_10	VCCK	VCCK	1.8V constant voltage supply for the chip core
60	C_11	GND	GNDK	Ground
61	B_11	GND	GNDO	Ground
62	C_10	VCCO	VCC30	3.3V constant voltage supply for the chip padring
63	A_11	RESA	RESET_SAMP	Signal to reset the S&A (Fig. 38)
64	B_10	CASC	CASCADE	Signal to connect the S&A sequentially (Fig. 38)
65	B_9	REPT	READ_Pixel_Test	Read signal for the test pixel comparator
66	A_10	R1PT	RST1_Pixel_Test	1st reset signal for the test pixel comparator
67	A_9	R2PT	RST2_Pixel_Test	2nd reset signal for the test pixel comparator
68	B_8	PUPT	PULL_Pixel_Test	Analog output of the test pixel
69	A_8	GND	GNDK	Ground
70	B_6	VCCK	VCCK	1.8V constant voltage supply for the chip core
71	B_7	VCCO	VCC30	3.3V constant voltage supply for the chip padring
72	A_7	GND	GNDO	Ground
73	C_7	VPIX	VPIX	Reset voltage of the pixel array photodiodes (Fig. 30)
74	C_6	S19	S19	20th bit of the last S&A (CASCADEOUT of Fig. 38)
75	A_6	S18	S18	19th bit of the last S&A (CASCADEOUT of Fig. 38)
76	A_5	S17	S17	18th bit of the last S&A (CASCADEOUT of Fig. 38)
77	B_5	S16	S16	17th bit of the last S&A (CASCADEOUT of Fig. 38)
78	C_5	S15	S15	16th bit of the last S&A (CASCADEOUT of Fig. 38)
79	A_4	VCCO	VCC30	3.3V constant voltage supply for the chip padring
80	B_4	GND	GNDO	Ground
81	A_3	GND	GNDK	Ground
82	A_2	VCCK	VCCK	1.8V constant voltage supply for the chip core
83	B_3	S14	S14	15th bit of the last S&A (CASCADEOUT of Fig. 38)
84	A_1	S13	S13	14th bit of the last S&A (CASCADEOUT of Fig. 38)
Ex.	C_3	NC		Not connected

Table 7: Pin assignment (second half).

Table 8 and Table 9 describe the most stringent timing requirements of the control signals based on post layout simulations, it can guide in the choice of a setup most adequate for testing.

- W/B. NO.: Is the number of the finger that the wire bond is attached to (Fig. 42).
- PIN NO.: Displays the coordinates of its corresponding pin (Fig. 41).
- Short Name: Name of the pins as included in Table 6 and Table 7.
- PAD Name: Indicates the name of the pad as included in the Layout design in Cadence.
- GND and V_{dd}: Indicate the time a signal spends at logic ‘0’ or ‘1’ respectively.
- Rise Fall Time: Indicates the time a signal needs to pass from GND to V_{dd} and vice versa.

W/B NO.	PIN NO.	Short Name	PAD Name	GND	Rise Time	Vdd	Fall Time
1	B_2	S12	S12			40ns(3.3V)	
2	C_2	S11	S11			40ns(3.3V)	
3	B_1	S10	S10			40ns(3.3V)	
4	C_1	GND	GNDK	Always			
5	D_2	VCCK	VCCK			Always (1.8V)	
6	D_1	VCCO	VCC30			Always (3.3V)	
7	E_3	GND	GNDO	Always			
8	E_2	S9	S9			40ns(3.3V)	
9	E_1	S8	S8			40ns(3.3V)	
10	F_2	S7	S7			40ns(3.3V)	
11	F_3	S6	S6			40ns(3.3V)	
12	G_3	S5	S5			40ns(3.3V)	
13	G_1	VPIX	VPIX			Always(~[1.1-1.6]V)	
14	G_2	GND	GNDK	Always			
15	F_1	VCCK	VCCK			Always (1.8V)	
16	H_1	S4	S4			40ns(3.3V)	
17	H_2	S3	S3			40ns(3.3V)	
18	J_1	S2	S2			40ns(3.3V)	
19	K_1	S1	S1			40ns(3.3V)	
20	J_2	S0	S0			40ns(3.3V)	
21	L_1	VCCO	VCC30			Always (3.3V)	
22	K_2	GND	GNDO	Always			
23	K_3	SEEX	SEED_EXIT			20µs(3.3V)	
24	L_2	GND	GNDO	Always			
25	L_3	VCCO	VCC30			Always (3.3V)	
26	K_4	VCCK	VCCK			Always (1.8V)	
27	L_4	GND	GNDK	Always			
28	J_5	RST1	RST1	20µs	1ns	40ns(1.8V)	1ns
29	K_5	RST2	RST2	20µs	1ns	20ns(1.8V)	1ns
30	L_5	READ	READ	50ns	1ns	20µs(1.8V)	1ns
31	K_6	RESE	RESET	60ns	1ns	20µs(1.8V)	1ns
32	J_6	VPIX	VPIX			Always (~[1.1-1.6]V)	
33	J_7	GND	GNDO	Always			
34	L_7	VCCO	VCC30			Always (3.3V)	
35	K_7	VCCK	VCCK			Always (1.8V)	
36	L_6	GND	GNDK	Always			
37	L_8	CEDI	CELL_DISABLE	20µs	1ns	20µs(3.3V)	1ns
38	K_8	CERS	CELL_RST	20µs	1ns	50ns(3.3V)	1ns
39	L_9	VSTR	IREF			Always (~[0.4-0.6]V)	
40	L_10	VREF	VREF			Always (~[1-1.5]V)	
41	K_9	SEED	SEED	50ns	1ns	50ns(3.3V)	1ns
42	L_11	SECL	SEED_CLK	50ns	1ns	50ns(3.3V)	1ns
43	K_10	VCCK	VCCK			Always (1.8V)	
44	J_10	GND	GNDK	Always			
45	K_11	GND	GNDO	Always			
46	J_11	VCCO	VCC30			Always (3.3V)	
47	H_10	CECL	CELL_CLK	20µs	1ns	50ns(3.3V)	1ns
48	H_11	CODI	COUNT_DISABLE	20µs	1ns	20µs(3.3V)	1ns
49	F_10	COCL	COUNT_CLK	10ns	1ns	10ns	1ns
50	G_10	CORE	COUNT_RESET	20µs	1ns	50ns(3.3V)	1ns

Table 8: Timing requirements of the signals associated to each pad (first half).

W/B NO.	PIN NO.	Short Name	PAD Name	GND	Rise Time	Vdd	Fall Time
51	G_11	REDY	READY	20µs	1ns	10ns(3.3V)	1ns
52	G_9	VPIX	VPIX			Always (~[1.1-1.6]V)	
53	F_9	VCCK	VCCK			Always (1.8V)	
54	F_11	GND	GNDK	Always			
55	E_11	VSPT	IREF_Pixel_Test			Always (~[0.4-0.6]V)	
56	E_10	VPPT	VPIX_Pixel_Test			Always (~[1.1-1.6]V)	
57	E_9	VRPT	VREF_Pixel_Test			Always (~[1-1.5]V)	
58	D_11	RSPT	RESET_Pixel_Test	60ns	1ns	20µs(3.3V)	1ns
59	D_10	VCCK	VCCK			Always (1.8V)	
60	C_11	GND	GNDK	Always			
61	B_11	GND	GNDO	Always			
62	C_10	VCCO	VCC30			Always (3.3V)	
63	A_11	RESA	RESET_SAMP	20µs	1ns	10ns(3.3V)	1ns
64	B_10	CASC	CASCADE	20µs	1ns	100ns(3.3V)	1ns
65	B_9	REPT	READ_Pixel_Test	50ns	1ns	20µs(3.3V)	1ns
66	A_10	R1PT	RST1_Pixel_Test	20µs	1ns	40ns(3.3V)	1ns
67	A_9	R2PT	RST2_Pixel_Test	20µs	1ns	20ns(3.3V)	1ns
68	B_8	PUPT	PULL_Pixel_Test			5ns(1.8V)	
69	A_8	GND	GNDK	Always			
70	B_6	VCCK	VCCK			Always (1.8V)	
71	B_7	VCCO	VCC30			Always (3.3V)	
72	A_7	GND	GNDO	Always			
73	C_7	VPIX	VPIX			Always (~[1.1-1.6]V)	
74	C_6	S19	S19			40ns(3.3V)	
75	A_6	S18	S18			40ns(3.3V)	
76	A_5	S17	S17			40ns(3.3V)	
77	B_5	S16	S16			40ns(3.3V)	
78	C_5	S15	S15			40ns(3.3V)	
79	A_4	VCCO	VCC30			Always (3.3V)	
80	B_4	GND	GNDO	Always			
81	A_3	GND	GNDK	Always			
82	A_2	VCCK	VCCK			Always (1.8V)	
83	B_3	S14	S14			40ns(3.3V)	
84	A_1	S13	S13			40ns(3.3V)	
Ex.	C_3	NC					

Table 9: Timing requirements of the signals associated to each pad (second half).

4.3 PRINTED CIRCUIT BOARD AND FIELD-PROGRAMMABLE GATE ARRAY

Based on the description of the prototype given in section 4.1 and the signal requirements presented in section 4.2, the Printed Circuit Board (PCB) that we designed for testing 'CSImager1' is as follows (Fig. 43):

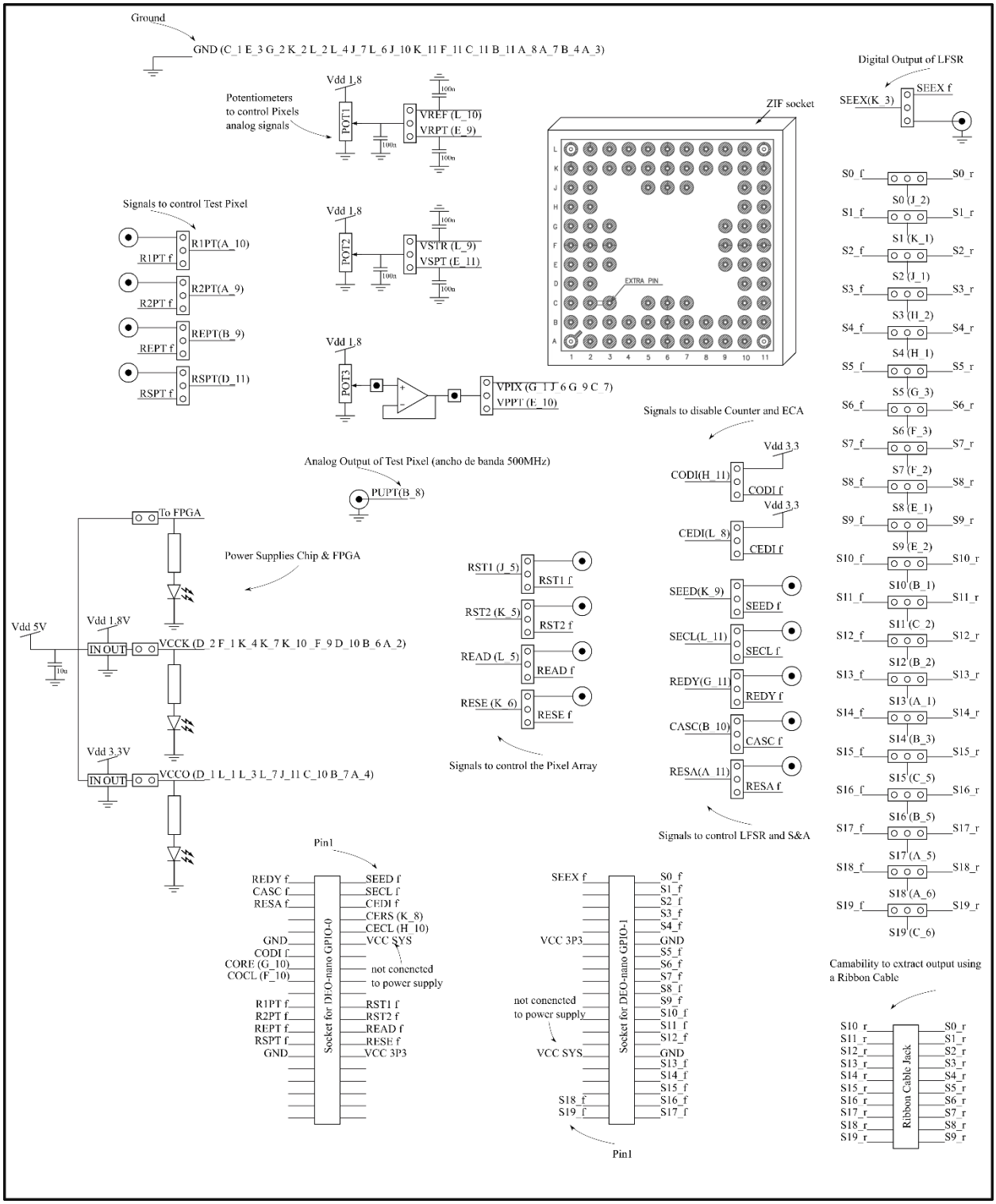

Fig. 43: Conceptual floorplan for the PCB design.

To design this PCB we planned to create two connections to each input and output of the chip that will be lodged in the Zero Insertion Force (ZIF) socket (Fig. 43). One of these connections will be sent to two parallel sockets onto which we will mount the extension headers of a Field-Programmable Gate Array (FPGA) (Fig. 43) while the others will be routed directly to accessible test points on the PCB. The first set of connections will be used to perform automated tests while the other can be attached to an exterior wave generator and/or an oscilloscope to perform tests that might exceed the FPGA capabilities.

The same design philosophy has been applied to the power source: it will be possible to power 'CSImager1' by using an external power source or by directly tapping the FPGA board. These redundancies are beneficial because, if any line proves to be damaged (short-circuited or open), we can use a mixed configuration to carry out the tests (as it was our case).

Only analog control signals were left out from this double implementation. Due to their nature and purpose (see section 4.1.2.7) we decided it was best to have full manual control over them by means of a set of potentiometers. Remember that, they just need to be fine-tuned for the appropriate environmental light conditions at the beginning of an experiment and then simply left alone for the duration of its duration.

The resulting PCB is shown in Fig. 44:

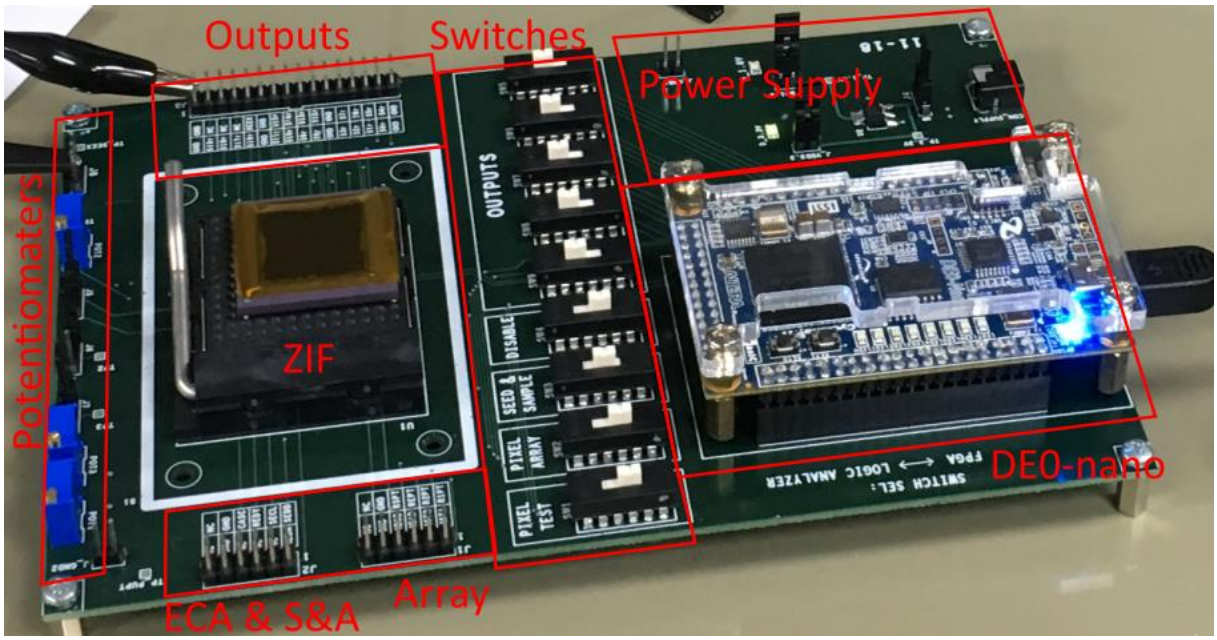

Fig. 44: The PCB used for our tests.

To the bottom right of Fig. 44 we can see the board (DE0-nano) attached to the PCB and at the centre of the left side the chip lodged in its ZIF socket. The switches between them are used to select if the inputs/outputs of the chip will reach the FPGA or the dedicated test points around

the ZIF socket. In the upper right part of this image there is the circuitry dedicated to the power supply, with independent jumpers to cut off the 3.3V needed to power the padring and the 1.8V needed to power the chip core. Here we can also find a third jumper that can be used to switch between jacks to which connect an external power source and the power offered by the board itself.

In the image above we can see a mixed configuration in which ECA, S&A, pixel array and power source are handled by the FPGA and the outputs of the chip are sent to the PCB test points (top left of Fig. 44).

The FPGA that we chose to conduct our tests is an Altera Cyclone IV (Fig. 45) which sports two 40-pin expansion headers on the long sides that we will use to access 'CSImager1', 8 LEDs and 2 push buttons that we can use to program tests divided in different phases and a clock frequency of 50MHz which might be a bit slow if we consider Eq. (49) but that for initial testing is more than enough. Among the 2 expansion headers we find 4 fixed 3.3V pins that can be used to power 'CSImager1'.

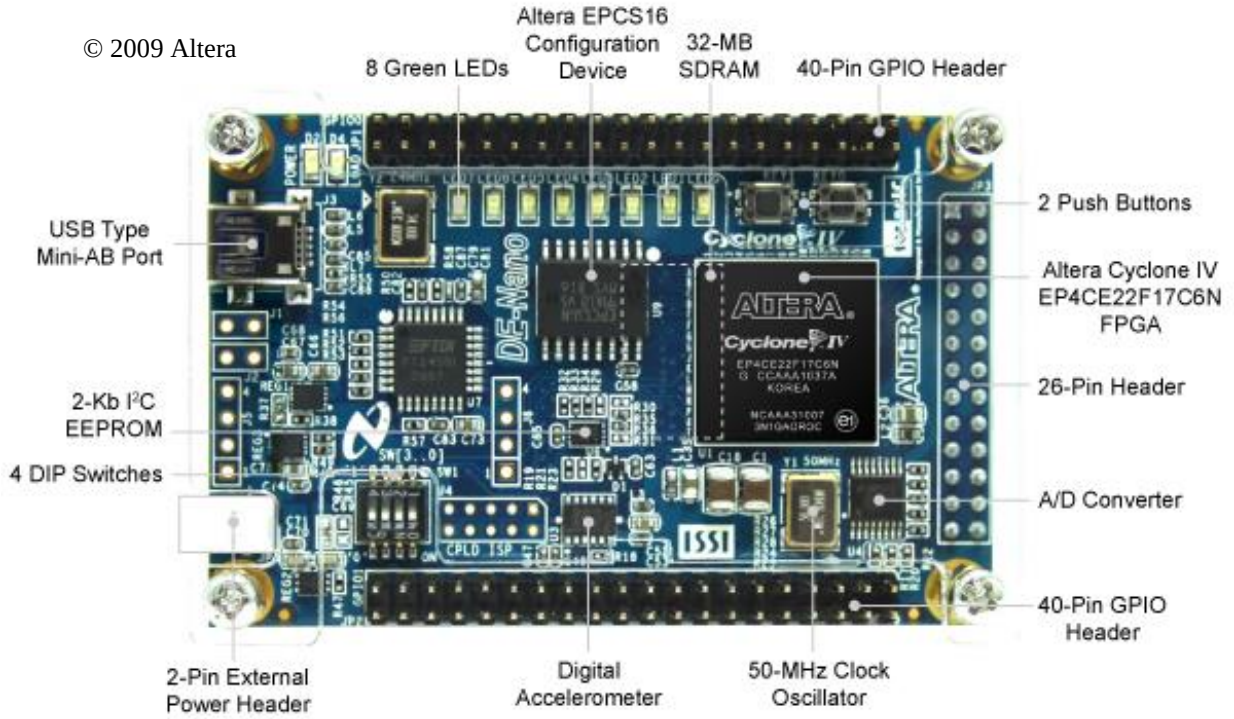

Fig. 45: Altera DE0-Nano FPGA © 2009 Altera.

4.4 TEST RESULTS ON CSIMAGER1

As mentioned in the circuit description, the correct operational state of many components can only be inferred from the CS-CIS 20 bit digital output. As we have described in section 4.1 care was taken to add the opportune control signals that disable parts of the prototype and allow us to study them separately. These precautions give us a certain amount of freedom when the time comes to perform tests on different parts of 'CSImager1' but the suggested order to conduct these experiments is the following:

- Test Pixel and Event Control unit
- Seed registers
- Sample & accumulate
- Counter
- Pixel array and measurement matrix

The proposed order is mainly due to the fact that the pixels are the only components with mixed signals in them, they are the core of our project, whose dynamic behavior can validate the feasibility of the theory so far exposed in a CMOS 0.18μm technology. Furthermore, the test pixel has both inputs and outputs totally independent from the reset of the prototype making it a good place to start testing since the rest of the circuit performance is yet an unknown.

The seed registers have a dedicated output pin to confirm that the seed is being introduced correctly. Signal SEEX (K_3) taps directly into the last of the 128 flip-flops loaded with the seed. This is another system that can be tested with relative ease and it is important to know if the seed is input correctly before starting any possible test on the array.

The S&A will be the third part to be tested because, since every experiment involving the remaining parts of the prototype deliver their outputs through the S&A, it would be logical to know if this component works properly before venturing into more complicated tests.

The counter has then been chosen as the fourth test because of its simplicity both in design and function and lastly, disabling the counter and knowing that the seed registers work correctly, the pixel array and measurement matrix will be tested almost at the same time since their workings are intertwined.

4.4.1 TEST PIXEL AND EVENT CONTROL UNIT

We will perform this test under the most demanding scenario presented by Eq. (47), following the suppositions made on Eq. (48) that led as to the range of frequencies expressed by Eq. (49) we will consider $C_{ph} = 60\text{fF}$ and a range of currents between 300pA and 30pA. For the reasons

above presented we have selected $V_{pix} - V_{ref} = 0.1V$ to time the pixel responses within a range of 2μs to 20μs. Through post layout simulations we have selected $V_{str} = 0.22V$ to set the duration of the pixel events to $t_e \sim 100ns$ so to make it proportional to the other signals controlled by the Verilog state machine loaded into the DE0-Nano's Altera Cyclone IV for this test.

This first test will also serve us as calibration of sorts to confirm if these analog signals are set properly: if the difference $V_{pix} - V_{ref}$ is not wide enough the pixel response would be too fast and vice versa if it is too large then response would be slow. In the same way, if V_{str} was too low or too high the event would not be registered by the output or its duration would simply be too long.

The test pixel output is accessible from an analog pad (pad 68 with coordinates B_8 of Table 4 and Table 5) and is transmitted to a test point directly onto the PBC. The list of signals that we need to control in order to operate the test pixel is the following:

- Pad 58 (D_11, RSPT): The reset signal of the test pixel;
- Pad 65 (B_9, REPT): The control signal that operates the transmission gate that starts the event generation (Fig. 32);
- Pad 66 (A_10, R1PT): The control signal needed to operate the transmission gate that charges the capacitor of the autozeroing comparator with V_{ref} (Fig. 32);
- Pad 67 (A_9, R2PT): The control signal needed to operate the transmission gate that cancels the offset of the inverter of the autozeroing comparator while its capacitor is charging with V_{ref} (Fig. 32).

This test is set up with a mixed configuration of the PCB: the control signals and the power supply are handled by the FPGA while the outputs are collected with an oscilloscope. Fig. 46 represents the state machine that has been loaded into the FPGA to perform this test and that reflects the timing diagram presented in Fig. 33. The actual Verilog code used for this experiment can be found in Appendix A.

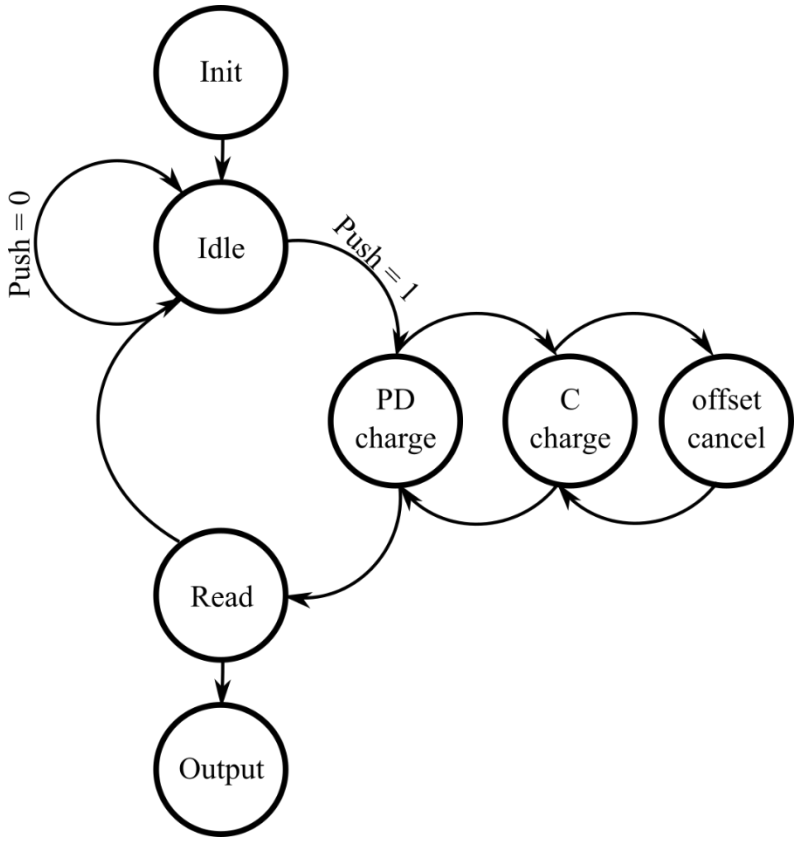

Fig. 46: State Machine for Test Pixel Control.

Upon initialisation all the digital control signals of the test pixel are set to logic ‘0’ with the exception of RSPT which is set to logic ‘1’ to disconnect the cathode of the photodiode from V_{cat} (Fig. 29). The state machine is set to idle while waiting for an external input.

The external input comes in the form of a simultaneous tap on the two push buttons of the DE0-Nano (Fig. 45); this action is represented in Fig. 46 by the variable “Push”. During state ‘PD charge’, flipping RSPT to logic ‘0’ enables the pixel reset (Fig. 29), the cathode of the photodiode V_{cat} charges at the desired voltage V_{pix}. While this is happening the connection between the cathode and the comparator is cut off by the transmission gate controlled with REPT (read signal in Fig. 32).

As the photodiode is charging, by enabling the transmission gate controlled by signal R2PT (state ‘C charge’) the capacitor in the autozeroing comparator is loaded with the chosen reference voltage (V_{ref}). Within this time frame, while the capacitor is reaching the desired reference, the transmission gate used to remove the inverter offset (R1PT of Fig. 32) is also enabled (state ‘offset cancel’). It is important that the activation of R1PT be contained in that of R2PT as shown in Fig. 33.

After this setup phase, both transmission gates are disabled and the capacitor is connected to the voltage of the cathode (state ‘Read’). This is achieved enabling the transmission gate

controlled by signal READ while switching off the reset transistor M1 (Fig. 29) so that the photodiode can start discharging with a speed proportional to the amount of light that it receives. After a fixed amount of time the state machine returns to idle. During this time frame if an events occurs than it is recorded by the oscilloscope. The allotted time is equal to the time that it would take for the counter (section 4.1.4) to reach '00000000', if no event has occurred during that time than the pixel is considered black.

Different tests have been carried out using different light conditions:

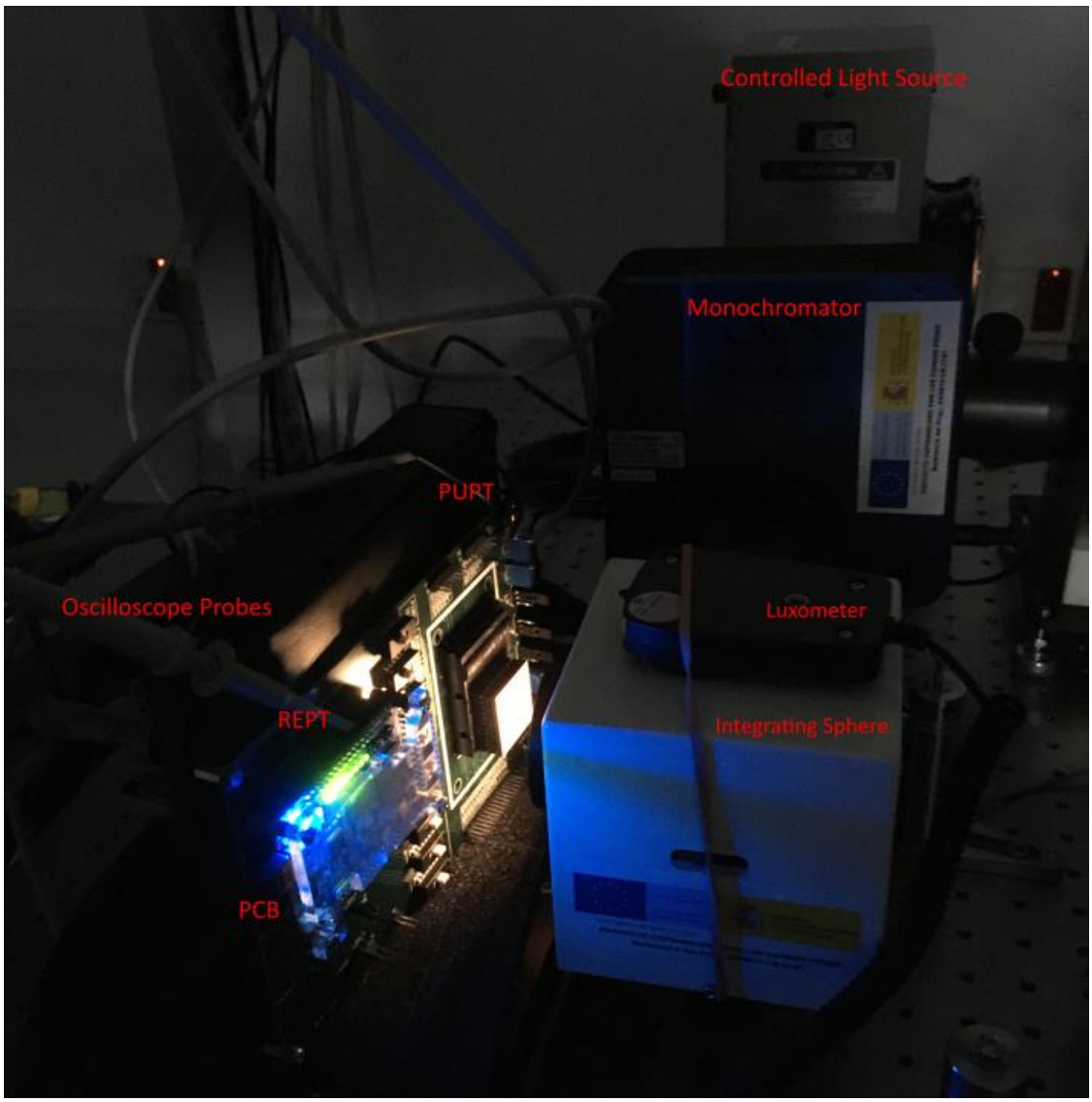

Fig. 47: Pixel test experiment setup.

We used a power controlled light source and a luxometer to provide 'CSImager1' with different light intensities ranging from 120lux to 1560lux (Fig. 47). The light was diffused using an integrating sphere. The monochromator between the light source and the sphere is not part of this experiment so it is set to a neutral position. 'CSImager1' mounted on the PCB was attached

to one of the outputs of the integrating sphere being the other attached to the luxometer. Here in Fig. 47 the PCB is slightly detached to offer a better view of the experimental setup.

Two probes of an oscilloscope are used to collect signals REPT (B_9) from a pin of the FPGA board and PUPT (B_8) from a test point on the PCB. The moment when REPT flips from logic '0' to logic '1' in state Read (Fig. 46) coincides with the moment in which the photodiode start discharging and when a pulse is shown on PUPT coincides with the moment when the event would be collected by the S&A. Setting the oscilloscope to stop when it detects the falling edge PUPT will give us a still picture of the time needed from when the photodiode starts discharging to when the pixel delivers the corresponding event (Fig. 48):

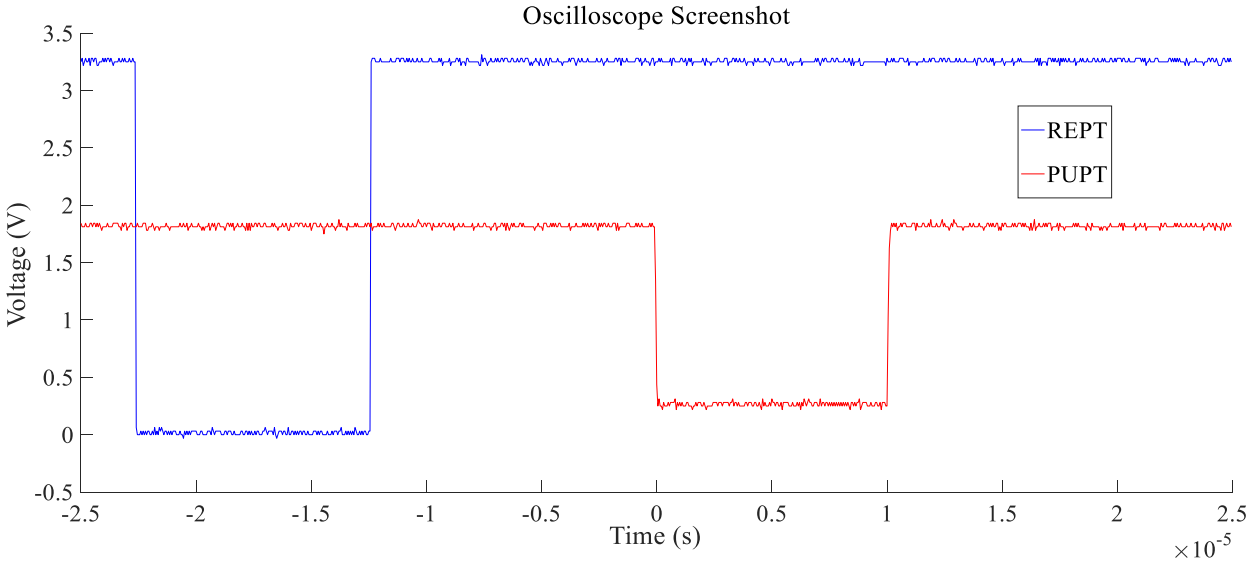

Fig. 48: Oscilloscope still image taken using a light intensity of 410lux.

Looking at Fig. 48 we can see that the width of the pulse shown in PUPT is proportional to that of REPT, this means that our estimate of V_{str} was correct, but, more importantly, we can see that the time that has passed between when the photodiode starts discharging (rising edge of REPT) and when the pulse appears on screen (Falling edge of PUPT) is about 12μs, which is about half the range that we had predicted. This means that, the time encoding of light intensity is carried out successfully by the pixel in the time that we had predicted theoretically. Please note that both events were stretched using a waiting function within the controller and reducing V_{str} so that they could have a duration proportional to the time that passes between them in order to better visualise the results on the screen of the oscilloscope. During normal operations these pulses would be few nanoseconds wide and thus trying to capture them in the same screenshot would be impossible.

By repeating this experiment with different light intensities and creating a graph of light intensity over response time we can express the characteristic curve that represents how our pixel

performs time encoding:

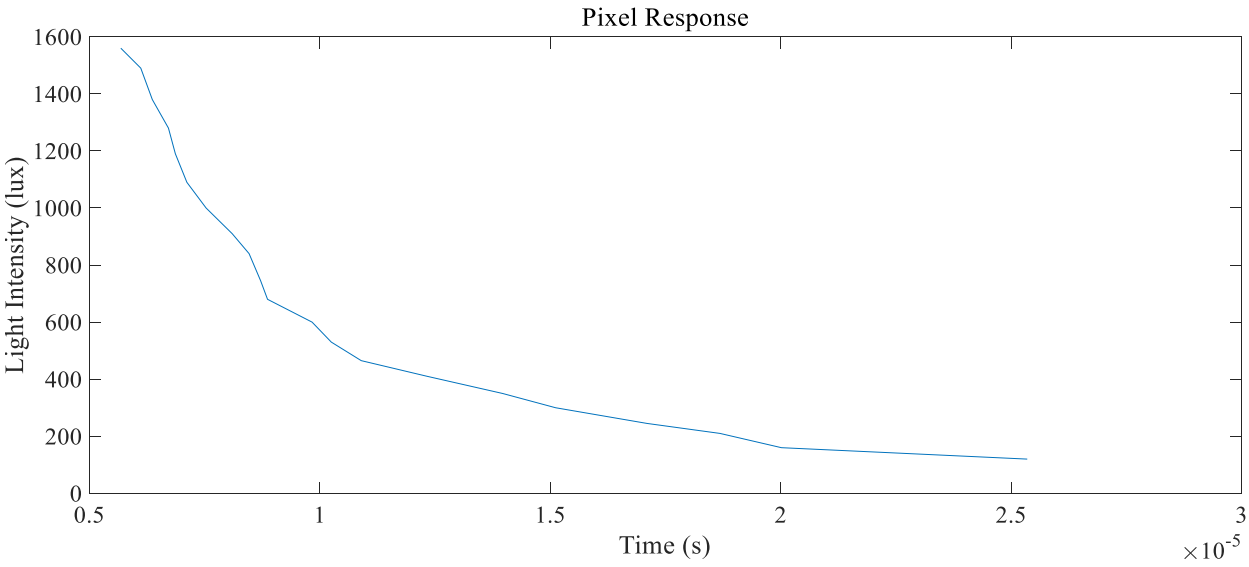

Fig. 49: Test Pixel Time response to varying light conditions.

4.4.2 SEED REGISTERS

In order to guarantee that reconstruction of the sampled image from the compressed samples delivered by our prototype is possible, it is necessary to confirm that the seed used to generate the measurement matrix is stored correctly: if this were not the case, the generated measurement matrix would not follow our predictions and the output delivered by 'CSImager1' would simply be undecipherable.

In 'CSImager1' the seed is loaded within a 128-bit shift register having each of its flip-flop directly embedded in a cell of the ECA (Fig. 28). Remember that this registers can also be used to replace the ECA as an alternate form of choosing which pixels to activate and which not (This is achieved by setting CEDI (Pad 37 coordinates L_8) to logic '1').

The list of signals that we need to tap in order to test the shift register is the following:

- Pad 41 (K_9, SEED): The input of the register;
- Pad 42 (L_11, SECL): The clock signal of the shift register that, at each double flip, is used to introduce sequentially the actual value of SEED into the register by shifting forward one step the content already stored (Fig. 28);
- Pad 23 (K_3, SEEX): this pad taps into the state of the last flip-flop of the register. After 128 clock cycles of SECL we aspect to see at this output the same sequence introduced in SEED simply delayed 128 steps.

Just like with the previous test, we chose a mixed set up for the PCB: the input signals and the power supply are handled by the FPGA while the output is probed with an oscilloscope. The

controller loaded into the FPGA for this experiment is a simple 2-bit counter with the inverse of the least significant bit (b_l of Fig. 50) connected to $SEED_{clk}$ (Fig. 28) and the most significant bit (b_m of Fig. 50) connected to $SEED_{in}$ (Fig. 28). Each time the least significant bit is set to logic '0' $SEED_{in}$ is switched from high to low or vice versa and each time it is set to logic '1' the current value of $SEED_{in}$ is pushed forward along the sequence of registers that hold the seed of the ECA.

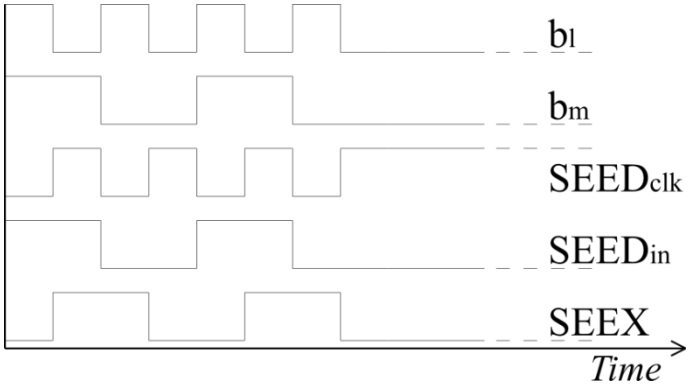

Fig. 50: Control signals for the ECA experiment.

The output of the last register is connected to a test point of the PCB (SEEX of Table 6) and, since what this controller creates is a simple train of pulses, SEEX should show the same trend as $SEED_{in}$ just delayed by 128 steps (the number of registers) and with a $\pi/2$ phase lag (remember that SEEDin switches when the least significant bit of the controller reaches '0' but it is pushed forward only after it returns to '1'). So, if we were to probe SEEX with an oscilloscope left continuously running, we should detect the same train found on the $SEED_{in}$ pin of the FPGA board as shown in Fig. 50. The FPGA clock frequency that we used to conduct this experiment was the maximum allowed by its specifications: 50MHz.

The setup of this experiment (Fig. 51) is much simpler than the previous one because there is no need to introduce a controlled light source and the rest of instruments that it required by the light beams to uniformly reach the active areas of the prototype (i.e. integrating sphere and luxometer).

After confirming that the output SEEX was indeed reflecting a train of pulses we proceeded with another experiment. This time, instead of using a steady train of pulses, we connected $SEED_{in}$ and $SEED_{clk}$ to the two buttons of the FPGA board so that we could use one of them to upload a logic '1' within the shift register and the other to upload a logic '0'.

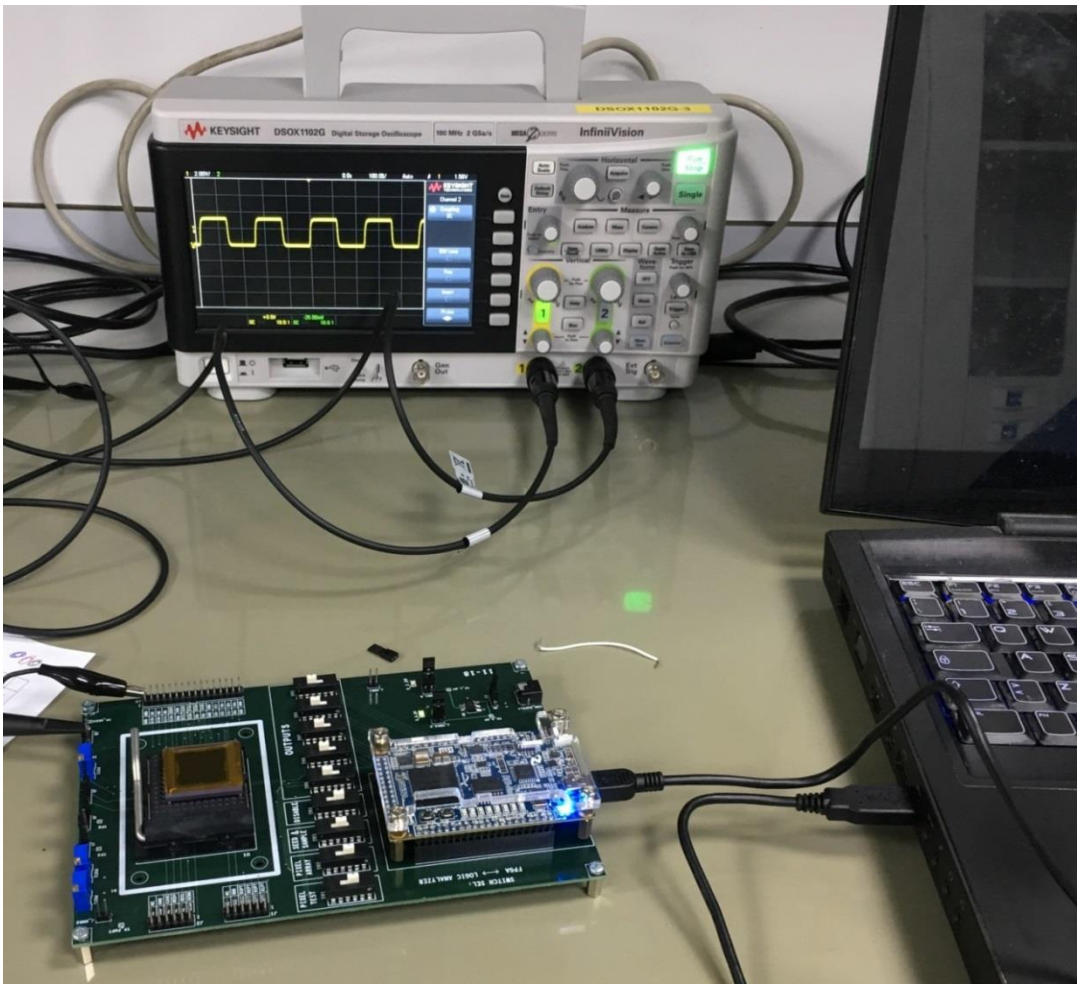

Fig. 51: Shift register test setup.

The LEDs provided by the FPGA board were used to keep track of the last inserted value (led1_A15 and led2_A13) and to check if at least 128 values had been uploaded (led8_L13). After 128 values were uploaded SEEX on the PCB started switching from logic ‘1’ to logic ‘0’ and vice versa according to the first values uploaded. Through this test we have indeed confirmed that a random sequence can be effectively used as seed without any risk of losing information and as such, assuming the correct implementation of the ECA, the prototype would deliver the compressed samples just as expected.

4.4.3 SAMPLE & ACCUMULATE

Having designed a sensor that performs in-pixel time encoding of light intensity, we are going to have to generate compressed samples by adding the contributions of the individual pixels in the digital domain through a time-to-digital converter. During normal operations a S&A (section 4.1.5) is used to timestamp the pixel outgoing pulses (section 4.1.2) with appropriate values generated by a counter (section 0). The circuitry of the S&A is also used to store the sum of these contributions until the end of the acquisition phase when it releases it to the CS-CIS output.

But, to perform tests that only involve studying the working of the S&A we will have to disconnect both the pixel array and the counter from its inputs.

This can be achieved in two steps, the first one consists in setting CODI (Pad 48 coordinates _11) to logic '1' in order to bypass the counter forcing its output to the fixed 8-bit logic word '00000001'. The second involves setting analog signals V_{ref}, around 1.8V and V_{pix} around 0V while also fixing digital signals READ and RST1 to logic '0' and RST2 to logic 1. This configuration makes it so that, inside each pixel, the photodiode cathode is permanently discharged and disconnected from the rest of the circuit and the autozeroing comparator is reduced to a simple inverter (Fig. 32) with a constant logic '1' as its input, thus rendering event generation impossible.

The list of signals that we have at our disposal to test the S&A is the following:

- Pad 51 (G_11, REDY): Named 'release' in Fig. 38 that can be used to insert artificial pulses in the Event handling circuitry of the S&A as described in section 4.1.5.4;
- Pad 63 (A_11, RESA): The signal used, in conjunction with REDY, to empty the registers where the S&A store the sum of all the events timestamps;
- Pad 64 (B_10, CASC): The signal used to connect all the S&A columns sequentially to add the column contributions together creating a compressed sample.

Since to test the S&A there is no need for a controlled light environment, the setup for this experiment is similar to that of the seed shift register (Fig. 51). As always we have chosen a mixed configuration for the PCB leaving in the hands of the FPGA to control of the three signals mentioned above and the powering of the prototype. We will use the probes of an oscilloscope or of a multimeter to tap the 20 bits of 'CSImager1' output to detect changes in the S&A.

The controller loaded into the FPGA this time around is rather simple, REDY and CASC are each connected to one of the two buttons of the FPGA board whereas RESA, the signal that resets the S&A array, is enabled by pressing the two buttons simultaneously. As explained in section 4.1.5.4 REDY can be used to emulate events coming from the pixel array, as such, pressing and depressing the button tied to this signal will simulate the arrival of said events. Since the last S&A column is directly connected to the 20 test points on the PCB that receive 'CSImage1' output and the counter is fixed at '00000001', each time this button is tapped, these test points should show a 20-bit binary sequence that is incremented by one unit per tap. As for the button connected to CASC, the signal that switches the S&A configuration from acquisition to avalanche in order to generate a compressed sample (section 4.1.5.2), tapping it will sum the contribution of the 64 S&A columns together. Note that, under the configuration for this experiment, each column of the S&A would increase by one unit each time REDY has been

tapped so that all the column will be loaded with the same number of events. For this reason, pressing CASC is in fact equal to multiplying the value of the last column times 64 (2^7). This will be reflected on the test points of the PCB as a movement of the 20-bit binary output sequence that will slide forward 7 steps.

Despite careful planning, in the beginning this experiment was not successful. After studying possible causes we have discovered that this was due to an oversight of the designer. Looking at how we designed the event handling unit in the S&A, we see that we have used an OR logic gate to combine the real events delivered by the array with signal release (Fig. 38) used to generate the fake events for this test. Since this part of the S&A was common to all bits and it sits at the junction of two large segments of the prototype, it was scarcely tested.

The oversight can be found in the fact that we considered the output bus as normally resting at logic '0' while jumping at logic '1' only during an event. This is the exact opposite of what happens in reality, since the output bus is kept at logic '1' by a pull-up PMOS transistor and a pixel that generate an event, pulls it near logic '0' by closing a stronger NMOS (M_2 in Fig. 29). Given this fact, an extra inverter should have been added to the bus before connecting it to the OR gate.

We found a roundabout solution to adequately perform the experiment at hand. First, we used the seed registers to input a seed which was zero for the most part except for the last element which was a one. We then set CEDI to logic '1' to bypass the ECA and select the pixels directly through the seed just set. In this way all the pixels of the bottom row would be active and ready to send an event. We then opened the starved inverters of the Event Control units (Fig. 34) by setting V_{str} at 0. In doing so, when a pixel generates an event this event will last indefinitely.

With this new setup we have initialized the FPGA and, before pressing its buttons, we have manually activates the potentiometer that controls V_{ref} to lower it value below 0.9V. This effectively switches the output of the inverter within the autozeroing comparator (Fig. 32) generating an endless event at the pixels of the bottom rows of the array that sets all output busses to logic '0' for the duration of the experiment.

This solution made it possible to test the correct functioning of the S&A using the test above detailed and the 20-bit output sequence functioned as described when we tapped REDY and CASC.

4.4.4 FURTHER TESTING

The presence of this design flaw marked the end of our planned tests. Since REDY not only generates simulated events, but it is also an active part of normal readout and reset operations

(see section 4.1.5.4), to further proceed with our tests we would need to be able to use it accordingly. But, as explained in the previous section REDY is rendered useless because of the OR gate that connects it with a signal that during readout and reset is a constant logic '1'.

One way to resolve this situation in order to continue testing this prototype and, if those test were successful, to use it to sample real images would involve two changes. Firstly we would need to renounce at the possibility of resetting the ECA so that we could use its seed registers to disable the pixel array in the same way that we did for the experiment described in the above section. And second a redesigning of the PCB would be necessary in order to automate the switching of V_{ref} and V_{str} using digitally controlled potentiometers. These potentiometers would be capable of changing these signals values at a speed comparable to that of the rest of the signals that control the prototype without disrupting its functionalities.

By taking these precautions we could operate 'CSImager1' following these steps:

1) Insert the seed inside the ECA and move its clock once so that the seed is stored in the rule 30 register (Fig. 28);
2) Replace the seed with a 128-bit word that is zero for the most part except for the last element which is a one;
3) Proceed through the acquisition phase normally;
4) Bypass the ECA by setting CEDI at logic '1';
5) Set Vref to Vdd and Vstr to GND using the aforementioned potentiometers to disable the pixel array;
6) Proceed through the formation of a compressed sample allowing the S&A to cascade;
7) Reset Vref and Vstr to the desired values and disable CEDI;
8) Return to step 3.

Table 10: New operational sequence for 'CSImager1'.

But, due to time constraints and considering the cost of designing a new PCB we decided to stop the tests. This was a good ending point because the studies of the test pixel and of the S&A were enough to confirm that PWM can be applied feasibly on CS-CIS.

CHAPTER 5
NON-RECURSIVE MOTION DETECTION FROM A STREAM OF COMPRESSED SAMPLES

Many algorithms that extract information from compressed samples have been successfully implemented. Some of these works tackle the problem of feature extraction trying to adapt existing algorithms to the compressed domain [Dave07]. Others focus on recursive algorithms that use trainable sparsifying dictionaries for object recognition [Nage09]. Among these works, background subtraction from compressed samples using reconstruction algorithms has been studied [Jian12]. In the past we presented a lightweight algorithm [Trev16] that aims at pre-process non-recursively the information contained in a set of compressed samples, in a context of video surveillance, to search for the presence of a movable object within a still frame. In this chapter, we adapt this non-recursive algorithm to detect movable object from the compressed samples generated by a CS-CIS that implements a binary measurement matrix of positives and negatives contributions ('1s' and '-1s') using a resettable PRNG and a differential readout method for the pixel array. The reason why we need these type of CS-CIS is because the simplicity of this algorithm is compensated by some requirements placed on the structure of the measurement matrix and the compressed samples that can be used. Specifically, this algorithm needs to operate on compressed samples obtained using a differential binary matrix that resets at every frame and the compressed samples delivered need to be split in two separate contributions for positive and negative matrix coefficients.

Differential readout methods that transform binary measurement matrices into matrices of ones and minus ones, such as the one presented in [Maji10] or the one analysed in section 3.2.1, although not eligible as stable solutions to reduce the bit size of compressed samples (see section 1.2), have an advantage over the use of standard binary matrices: since all the pixels are multiplied by coefficients that are either positives or negatives (but never zero), every compressed sample that they generate includes information on all the pixels of the CS-CIS. If we consider $\mathbf{\Phi}^{+}$, the conventional binary measurement matrix generated using a PRNG, we can

represent the transformation that a differential readout operates on it as:

$$\boldsymbol{\Phi} = \boldsymbol{\Phi}^{+} - \boldsymbol{\Phi}^{-} \tag{50}$$

being $\boldsymbol{\Phi}$ the new measurement matrix to be used for reconstruction where $\boldsymbol{\Phi}^{-}$ is:

$$\boldsymbol{\Phi}^{-} = \mathbf{J} - \boldsymbol{\Phi}^{+} \tag{51}$$

being $\mathbf{J} \in \mathbb{R}^{M \times N}$ a matrix of ones, where M is the number of compressed samples and N that of the pixels. Substituting Eq. (50) inside Eq. (7) we can reformulate the sampling process as:

$$\begin{bmatrix} \mathbf{y}^{+} \\ \mathbf{y}^{-} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}^{+} \\ \boldsymbol{\Phi}^{-} \end{bmatrix} \mathbf{x} \tag{52}$$

where the set of compressed samples extracted from the imager will be:

$$\mathbf{y} = \mathbf{y}^{+} - \mathbf{y}^{-} \tag{53}$$

Let us consider that, within a CS-CIS, the contribution of positive and negative pixels can be separated and A/D converted in two different sums, using for instance the PFM architecture of Fig. 23 but with two separate counters. In this case it would be possible to exploit the linearity of Eq. (52) by implementing a reset feature on the PRNG (such as the one presented in Fig. 28), to detect movement over a fixed background using [Trev16].

5.1 Non-recursive Motion Detection

The difference of two sets of compressed samples, $\mathbf{y}_k$ and $\mathbf{y}_{k-1}$ from two consecutive frames $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, taken by a CS-CIS that resets its measurement matrix $\boldsymbol{\Phi}$ after every frame, equals the set of compressed samples $\bar{\mathbf{y}}$ obtained by sampling $\bar{\mathbf{x}}$, if $\bar{\mathbf{x}}$ is the difference of these two frames:

$$\bar{\mathbf{y}} = \mathbf{y}_k - \mathbf{y}_{k-1} = \boldsymbol{\Phi}(\mathbf{x}_k - \mathbf{x}_{k-1}) = \boldsymbol{\Phi}\bar{\mathbf{x}} \tag{54}$$

If the background of a scene was fixed, such is the case in some surveillance cameras setups, a pixel by pixel difference of two consecutive frames would return values different from zero only for those pixels that have changed over time. For this reason $\bar{\mathbf{x}}$ would only contain information regarding objects that have moved over the time that has passed between those frames. Given the linearity of Eq. (54), this same conclusion is valid when we apply it to the difference between sets of compressed samples as long as these sets have been taken using the exact same measurement matrix (hence the need for a reset of the PRNG after every frame).

The results from Eq. (52) and Eq. (54) can be exploited to pre-process these compressed samples. Combining these two equations we obtain:

$$\begin{bmatrix}\bar{\mathbf{y}}^{+}\\ \bar{\mathbf{y}}^{-}\end{bmatrix} = \begin{bmatrix}\mathbf{y}_k^{+} - \mathbf{y}_{k-1}^{+}\\ \mathbf{y}_k^{-} - \mathbf{y}_{k-1}^{-}\end{bmatrix} = \begin{bmatrix}\mathbf{\Phi}^{+}\\ \mathbf{\Phi}^{-}\end{bmatrix}(\mathbf{x}_k - \mathbf{x}_{k-1}) = \begin{bmatrix}\mathbf{\Phi}^{+}\\ \mathbf{\Phi}^{-}\end{bmatrix}\bar{\mathbf{x}} \tag{55}$$

Now let us consider the structure of the data generated by a CS-CIS that implements a resettable measurement matrix and a differential readout when it samples two consecutive frames and we then subtract one from the other. Remember that, each compressed sample $\bar{\mathbf{y}}_i$, is a combination of all pixel by pixel differences generated by the same row of $\mathbf{\Phi}$:

$$\begin{bmatrix}\bar{\mathbf{y}}_i^{+}\\ \bar{\mathbf{y}}_i^{-}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_i^{+}\\ \boldsymbol{\varphi}_i^{-}\end{bmatrix}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{56}$$

If the pixels that contribute to $\bar{\mathbf{y}}_i^{+}$ or $\bar{\mathbf{y}}_i^{-}$ did not contain any changes, i.e. no object appears to them or disappears from them, then their value will be zero because the subset of $\mathbf{x}_k$ that they contain equals the corresponding subset in $\mathbf{x}_{k-1}$. Likewise, if they differ from zero then something has changed in those subsets. The higher the absolute value of $\bar{\mathbf{y}}_i^{+}$ or $\bar{\mathbf{y}}_i^{-}$ the higher is the change present in the pixels selected by $\boldsymbol{\varphi}_i^{+}$ or $\boldsymbol{\varphi}_i^{-}$ between the two consecutive frames.

Based on these elements it is possible to create M contribution vectors $\mathbf{c}_i^{+} \in \mathbb{R}^N$ by multiplying each row of the measurement matrix, $\mathbf{\Phi}^{+}$, by the corresponding contribution in $\bar{\mathbf{y}}^{+}$:

$$\mathbf{c}_i^{+} = \boldsymbol{\varphi}_i^{+}\bar{\mathbf{y}}_i^{+} \tag{57}$$

Notice that each $\mathbf{c}_i^{+}$ is a vector that has the same size as the original frame N. All of its non-zero elements, which correspond to the positions of pixels selected by row $\boldsymbol{\varphi}_i^{+}$, will contain $\bar{\mathbf{y}}_i^{+}$. If an object moving within the frame appears in a spot captured by a pixel chosen by $\boldsymbol{\varphi}_i^{+}$, that object disappears from a pixel included in $\boldsymbol{\varphi}_i^{-}$. Which means that $\bar{\mathbf{y}}_i^{-}$ will also depart from zero. What is more is that $\boldsymbol{\varphi}_i^{+}$ together with $\boldsymbol{\varphi}_i^{-}$ contain the contributions of all the pixels of the frame, but no contribution belonging to $\boldsymbol{\varphi}_i^{+}$ will be in $\boldsymbol{\varphi}_i^{-}$, nor vice versa. As we have done for $\mathbf{\Phi}^{+}$ It is possible to define other contribution vectors $\mathbf{c}_i^{-} \in \mathbb{R}^N$ from $\mathbf{\Phi}^{-}$ and $\bar{\mathbf{y}}^{-}$:

$$\mathbf{c}_i^{-} = \boldsymbol{\varphi}_i^{-}\bar{\mathbf{y}}_i^{-} \tag{58}$$

Adding together these two set of vectors it is possible to obtain $\mathbf{c}_i \in \mathbb{R}^N$. These vectors $\mathbf{c}_i$ will contain elements that are either $\bar{\mathbf{y}}_i^{+}$ or $\bar{\mathbf{y}}_i^{-}$. The pixels selected by $\boldsymbol{\varphi}_i^{+}$ and $\boldsymbol{\varphi}_i^{-}$ have been chosen randomly. The number of pixels selected by them, even if very well balanced (section 2.5), is also random. This means that each pair $\boldsymbol{\varphi}_i^{+}$ and $\boldsymbol{\varphi}_i^{-}$ will generate a different contribution vectors

$\mathbf{c}_i$. It is worth nothing that if an object moves to subset of the image contained in $\boldsymbol{\varphi}_i^+$ from outside the frame, even if $\bar{\mathbf{y}}_i^+$ differs from zero, $\bar{\mathbf{y}}_i^-$ will still be zero. For that reason only full $\mathbf{c}_i$ will contain information on the motion of an object. A global contribution vector $\mathbf{c} \in \mathbb{R}^N$ can be generated by superposing the contribution of all $\mathbf{c}_i$ where both $\bar{\mathbf{y}}_i^+$ and $\bar{\mathbf{y}}_i^-$ are different from zero:

$$\mathbf{c} = \sum_i \frac{1}{H} (\boldsymbol{\varphi}_i^+ \bar{\mathbf{y}}_i^+ + \boldsymbol{\varphi}_i^- \bar{\mathbf{y}}_i^-) \quad (\forall i \in \bar{\mathbf{y}}_i^+ \neq 0 \vee \bar{\mathbf{y}}_i^- \neq 0) \tag{59}$$

being H the number of full vectors $\mathbf{c}_i$. The elements of $\mathbf{c}$ are linear combinations of the positive and negative contributions for each compressed sample. Each pixel will be accounted for the same exact number of time as any other. This means that each element of $\mathbf{c}$ is directly comparable to the others. Notice that this is true only because of $\boldsymbol{\varphi}_i^+$ and $\boldsymbol{\varphi}_i^-$ together address all the pixels of the frame.

The higher the absolute value associated to a given element of $\mathbf{c}$ the greater the contribution of its corresponding pixel will be. In a sense, the maxima and minima in $\mathbf{c}$ reflect those pixels where change happened. For that reason it is possible to associate those maxima and minima with the positions that the moving objects have occupied in the two consecutive frames.

Eq. (59) not only returns information on the presence of an object moving on a fixed background in two consecutive frames, it also gives information on the direction of that motion, i. e. where the object was and is likely to be. And last it is important to notice that none of the steps taken for the generation of $\mathbf{c}$ is recursive and, being this method not an optimisation of any sort, its accuracy is simply proportional to the amount of compressed samples employed.

5.2 Performance of the Non-recursive Motion Detector

The performance of this method was studied creating various synthetic videos of 50 frames of 64×64 pixels each. Each video has a black background and one or more moving objects represented by 3×3-pixel white squares. Beside the number of objects another variable that we considered was the amount of compressed samples extracted. As a 64×64-pixel frame has a total amount of pixel of 4096, we took sets of samples reaching a compression ratio that ranged from 1/2, with 2048 compressed samples up to 1/32 with 128 compressed samples, halving the total amount of samples taken at every step.

We compared each maxima and minima of the global contribution vector $\mathbf{c}$ obtained with our method with the difference of the original frames from which the compressed samples were derived. We also used the NESTA algorithm [Beck11] on the same sets to compare the

performance of our method to one of the most effective convex optimization reconstruction algorithms currently presented in literature.

Fig. 52 represents one example of our analysis. In this particular setup we used three moving objects and a set of 1024 compressed samples i. e. 1/4 compression. The top left image of Fig. 52 represents the difference of the original frames from which 1024 compressed samples and 1024 dual compressed samples have been simultaneously extracted. The top right image of Fig. 52 represents the reconstruction of the difference of said frames by applying a NESTA convex optimization algorithm over the set of compressed samples differences $\bar{\mathbf{y}}$. This difference has been further thresholded to remove low contributions and possible noise. This has been done to easily compare it to the maxima and minima of the weight vector. It is possible to see that it resembles closely the difference of the two original frames shown in the top left image of Fig. 52. Lastly the bottom image of Fig. 52 represents the location of the maxima (white) and minima (black) of the contribution vector produced by our proposed algorithm. These contributions have been scaled to fit a greyscale representation to easily compare them graphically with the original difference as well.

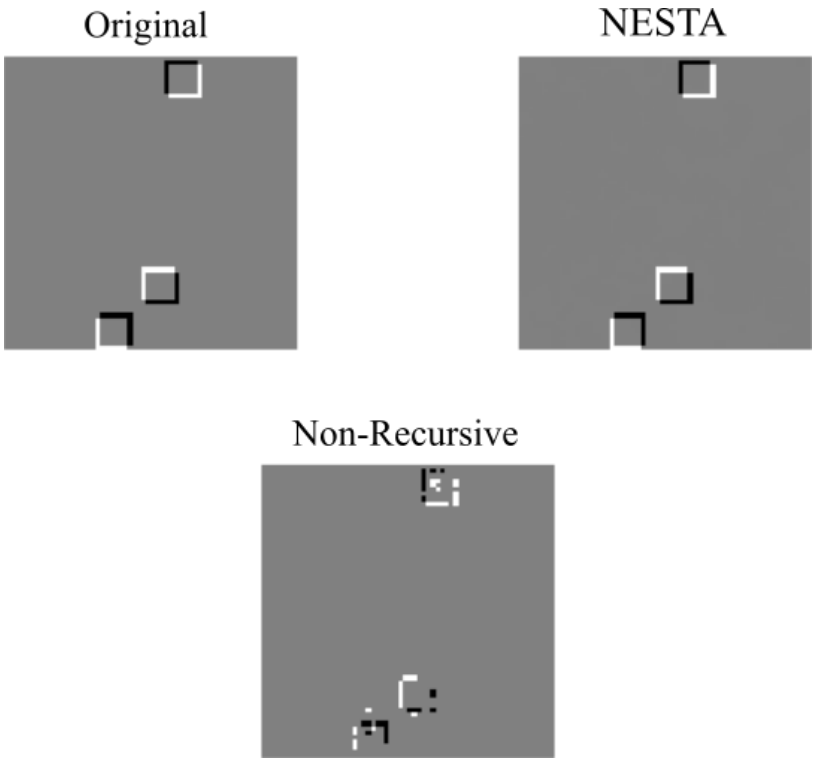

Fig. 52: (Top Left) Difference of two original 64 × 64-pixel frames; (Top Right) Filtered difference of two NESTA reconstructed 64 × 64-pixel frames; (Bottom) Maxima and Minima of the contributions extracted from two sets of compressed samples following our method.

To establish the reliability of this method we considered the pixel by pixel root mean square error (RMSE) of the scaled contribution vector defined in Eq. (59) using the original frame

differences as ground truth Fig. 53. The values that the RMSE of our method took were then compared to the RMSE through NESTA reconstruction using the same ground truth and the same set up. We recorded these values while varying the amount of objects moving in the scene and the amount of compressed samples taken.

Comparing the performance of these two methods movement it is possible to see that traditional reconstruction derives better results over our proposed methodology in all cases of study. This was to be expected because relying on convex optimization leads better results than relying on a method whose strength is based only on the sheer amount of samples taken.

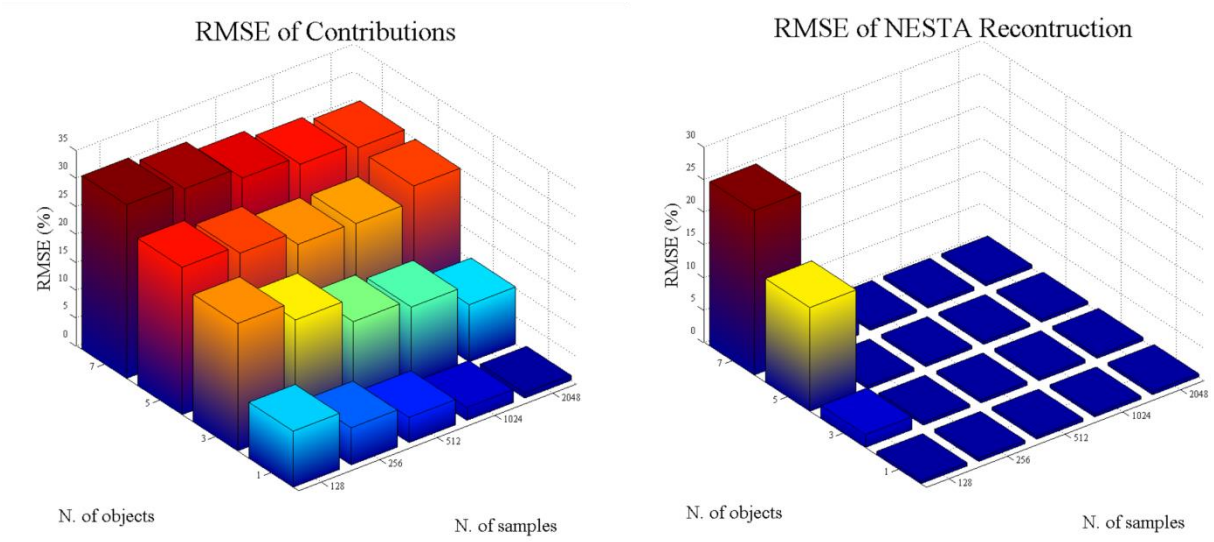

Fig. 53: (Left) RMSE of our method (% of full signal range); (Right) RMSE of NESTA reconstruction (% of full signal range).

Even if that is the case it is worth noting that NESTA reconstruction fails in returning acceptable results in extreme cases i. e. high number of objects and low number compressed samples. While trying to target specific information within the compressed samples thus not following a conventional reconstruction technique may deliver worse reconstruction errors it is also true that it can benefit from faster processing times opening the possibility to new applications of CS.

We have run the whole procedure on an Intel core i7-3740QM running at 2.7GHz having 24GB of RAM with an SSD. Comparing the time it took to perform the two simulations Fig. 54 it is possible to see that the longest time our method took to estimate the position of a moving object was 2 to 3 orders of magnitude lower than the time it took for NESTA to deliver its results.

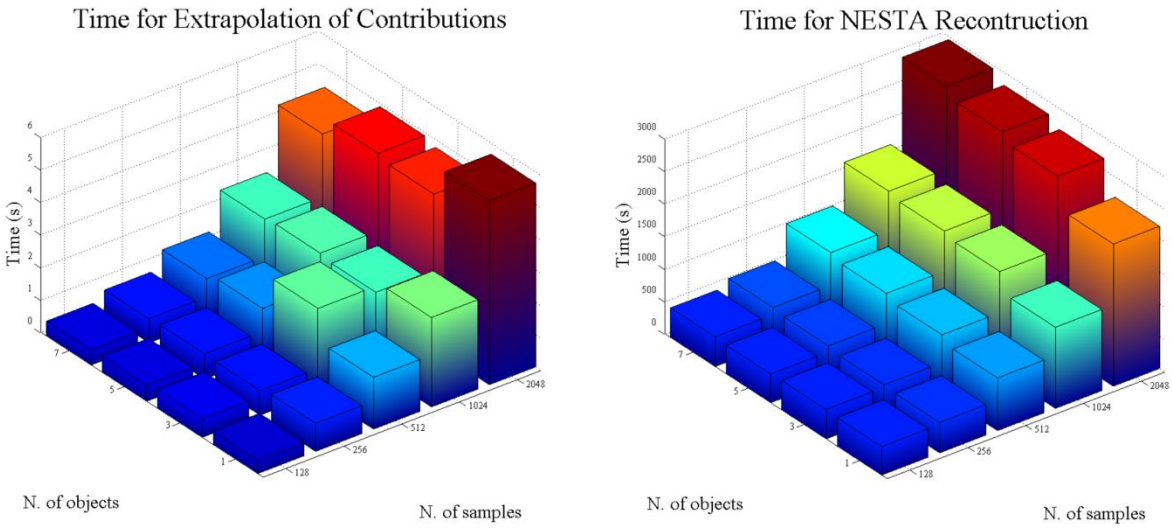

Fig. 54: (Left) Time needed for maxima and minima extraction (seconds), (Right) Time needed for NESTA reconstruction (seconds).

Please, take note that in Fig. 54 the time scale of the two graphics is completely different, whereas our algorithm takes mere seconds to detect the presence of a moving object, performing a reconstruction through NESTA algorithm takes almost five minutes (three orders of magnitude more than our algorithm).

Considering that the videos we were reconstructing had a total of 50 frames, in most cases it would have been possible to analyse said videos while streaming the results in real-time (at least 24 fps).

CHAPTER 6
TERNARY MEASUREMENT MATRICES BY MEANS OF CLASS III ECA

Through the use of PSD (section 2.3) and Density analysis (section 2.5) we have been able to establish a theoretical background to ascertain that rule-30 ECA and its mirrored and complementary rules are good approximations of a binary random measurement whose elements are extracted from a Bernoulli distribution having probability $P = 0.5$. We have also studied the possibility of avoiding the requirement of a wide dynamic range in the analog domain that usually befall compressed samples by using PWM (section 3.2.2) and we have demonstrated its feasibility in a CMOS 0.18μm 1P6M technology by testing a pixel designed following these principles (section 4.4.1).

Despite all these efforts and results, the RIP demonstration on random binary matrices presented in section 2.2 still poses a huge limitation cast upon the performance of CS-CIS imagers that use PRNG. To improve this result we introduce a new method for the generation of a hardware-friendly measurement matrix that takes into account both technology limits and quality of the resulting matrix. We propose a differential pixel readout system to recursively create ternary measurement matrices in a row by row fashion. For a fixed number of coefficients, such is the case of CS-CIS pixel arrays, the resulting matrices present smaller coherence than their binary counterparts thus improving the RIP sparsity order k, Eq. (21), and diminishing reconstruction errors. Similar results have been obtained using binary matrices with binary correlations between columns applied to CS in [ShuT16].

Ternary measurement matrices can be generated by delivering two sets of driving signals to each pixel: one used to define the coefficient value, either ‘0’ or ‘1’, and the other used to define the sign of the contribution, either positive or negative. As mentioned in the introduction, since we can consider random row and column selection as the multiplication of two random binary variables and because the multiplication of random variables still delivers random outputs, if one set of row/column selectors (or driving signals) can be used to produce a binary matrix having

one bit per coefficient, we could use two sets to produce a ternary matrix that needs two bits per coefficient.

If each pixel receives two driving signals, it would be possible to use one of them to determine if the pixel took part in a compressed sample and, in case it did, use the other to select the sign of its contribution. Since recent CS-CIS examples [Oike13], [Dadk15], [Leit18] have shown that, for an $N \times N$ pixel arrays, as few as $N_B = 2N$ bits of information are needed to generate binary measurement matrices by means of PRNG on-chip, the amount of resources (number of transistors) needed to incorporate the bits of information needed to generate a ternary matrix, $N_T = 4N$, would still be implementable in a CS-CIS design. Each coefficient of the resulting matrix would have a probability distribution of:

$$\begin{cases} \mathrm{P}_{+1} = 0.25 \\ \mathrm{P}_{0} = 0.5 \\ \mathrm{P}_{-1} = 0.25 \end{cases} \tag{60}$$

where P_{-1} represents the probability that a pixel has of contributing negatively in a compress sample, P_{+1} represents the probability that a pixel has of contributing positively and P_0 represents the probability that a pixel has of not contributing at all. Furthermore, as per what exposed in section 1.2 while introducing the concept of block based CS, a ternary measurement matrix would have a positive, even if not reliable, effect on the limit imposed by Eq. (11) on the amount of bits required to represent a compressed sample.

In hardware, the sign associated to a pixel contribution can be implemented by means of a differential readout system that routes the output of the pixels through one of two output lines [Maji10]. These lines, outside of the pixel array can then be digitized separately or used as input of differential circuits, such as analog subtractors or transimpedance amplifiers. Using a differential readout system to divide the pixel contribution to different output lines to be treated separately would in fact reduce the amount of bits needed for ADC thus relaxing the converter design parameters.

Splitting the pixel contributions in two separate sets could also be used to reduce the operational frequency of a CS-CIS that employed PFM to encode light intensity in a train of pulses and then performed time-to-digital conversion using simple counters (section 3.2.1). This could render PFM a viable alternative to the PWM implementation adopted for our prototype. Remember that we chose PWM over PFM simply because the excessive amount of pulses generated by a PFM pixel array was impossible to manage with the maximum frequency achievable by a CMOS 0.18μm 1P6M technology Eq. (42). If the frequency of the pulses were to

diminish, both modulations could be used to achieve de desired amount of bit to describe a compressed sample following Eq. (11).

6.1 PERFORMANCE OF THE GENERATED TERNARY MEASUREMENT MATRICES

We will evaluate the performance of these matrices, compared to binary measurement matrices obtained through ECA and LFSR via a suite of simulations carried out on 8-bit grayscale standard images of size 512×512 shown in Fig. 55, from left to right Lena, Boat, Livingroom, and Mandrill.

Fig. 55: Test images.

The Peak Signal-to-Noise Ratio (PSNR) is chosen as objective quality measure for our experimental framework. PSNR is an approximation to human perception of reconstruction quality from lossy compression. When used for this purpose the original data is considered the signal and the error introduced by compression is the noise:

$$\mathrm{PSNR} = 10\log_{10}\frac{\max(\mathbf{x})^2}{\frac{1}{MN}\|\mathbf{x}-\hat{\mathbf{x}}\|_2} \tag{61}$$

being $\max(\mathbf{x})$ the highest possible pixel value. The simulations were carried out in MATLAB. For each image, due to the random nature of the generated measurement matrices, 30 trials are performed and the average PSNR is computed.

The recovery of the sampled image is carried out using a Block-based Compressive Sampling Smoothed-Projected Landweber (BCS-SPL) [Mun09] reconstruction algorithm with Singular Value Decomposition (BCS-SVD) [Akb118] able to provide a fast and accurate image reconstruction from compressed samples obtained using PRNG generated measurement matrices, [Akb218], [Trev20].

We performed BCS sampling, using blocks of size 32×32, i.e. $N = 1024$. When very large images are involved dividing the pixel array into smaller sub-arrays that can be digitized independently becomes necessary for two reasons. It helps the implementation of practical measurement matrices in CS-CIS and it reduces software resources needed during

reconstruction. Usually fast and accurate reconstruction algorithms opt for small blocks, typically 8×8. On the other hand, blocks that are too small could potentially deteriorate the quality of the sensor introducing asymmetries within the pixel array or complicate its design. We chose a block size of 32×32 as a compromise between the two divergent necessities, and after having demonstrate that PWM can effectively be used to reach the desired dynamic range of the collected compressed samples (section 3.2.2).

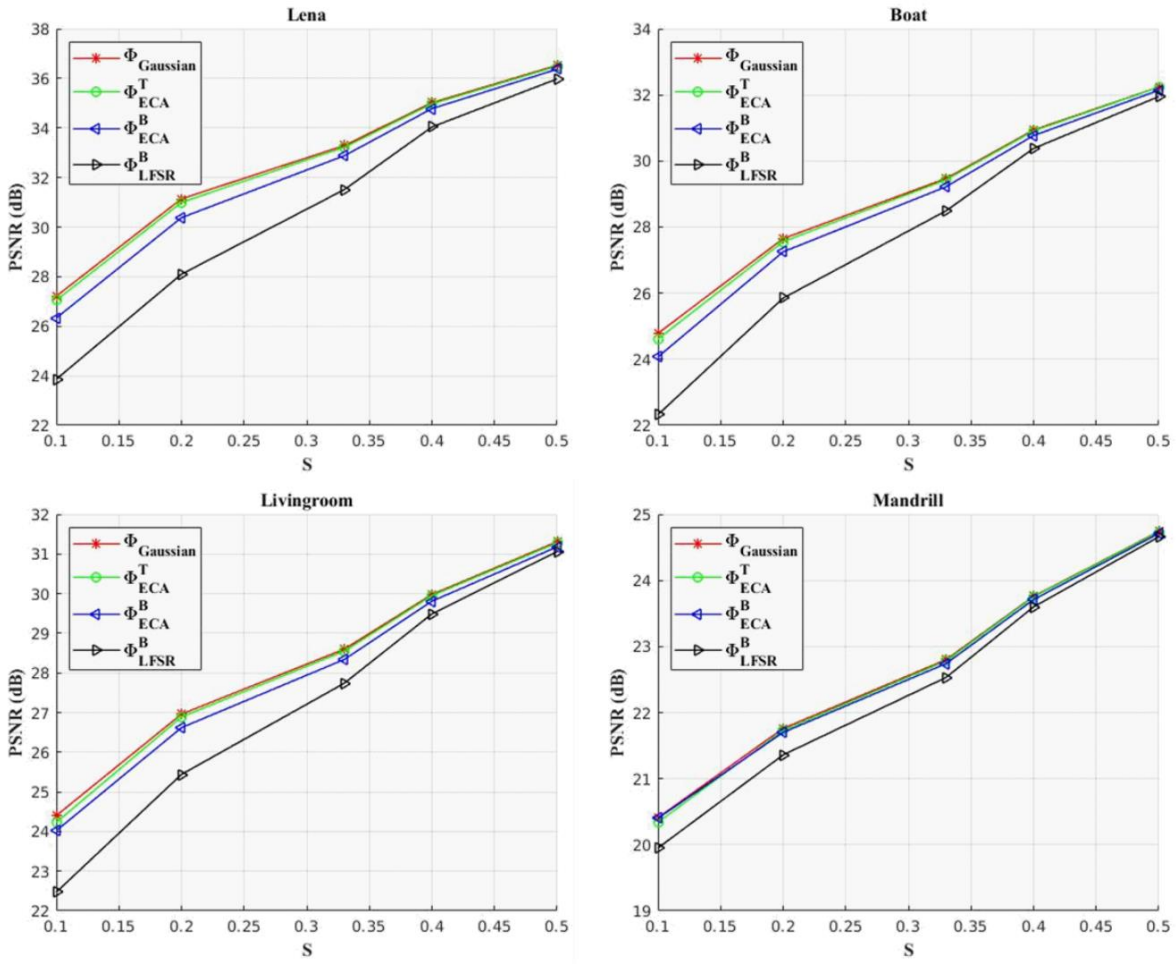

Fig. 56: PSNR of the reconstructed images as a function of subrate (S).

The performance of the proposed ternary matrix based on ECA, $\mathbf{\Phi}^{\mathrm{T}}_{\mathrm{ECA}}$, is compared to the performance of the pseudo random measurement matrices based on both LFSR, $\mathbf{\Phi}^{\mathrm{B}}_{\mathrm{LFSR}}$, and ECA, $\mathbf{\Phi}^{\mathrm{B}}_{\mathrm{ECA}}$. In addition, as a reference, we have included the Gaussian random matrices, $\mathbf{\Phi}_{\mathrm{Gaussian}}$, whose entries are randomly selected from a normalised Gaussian distribution.

Fig. 56 shows the quality of reconstructed images in terms of PSNR as a function of subrate, S, varying from 0.1 to 0.5. An interesting observation is that ternary matrices $\mathbf{\Phi}^{\mathrm{T}}_{\mathrm{ECA}}$ outperform binary matrices almost reaching the performance of Gaussian random matrices $\mathbf{\Phi}_{\mathrm{Gaussian}}$. Depending on the subrate, on average, the ternary matrices improve the PSNR of the

reconstructed images from 0.08 dB to 0.35 dB when compared with the binary measurement matrices based on ECA, $\mathbf{\Phi}_{\mathrm{ECA}}^{\mathrm{B}}$, and from 0.27 dB to 1.90 dB when compared with the binary measurement matrices based on LFSR, $\mathbf{\Phi}_{\mathrm{LFSR}}^{\mathrm{B}}$. Finally, it should be noted that the ternary matrices, $\mathbf{\Phi}_{\mathrm{ECA}}^{\mathrm{T}}$, compete with the performance of the Gaussian random matrix, $\mathbf{\Phi}_{\mathrm{Gaussian}}$, which is an optima theoretical measurement matrix.

To show that the proposed ternary measurement matrices are compatible with most of the robust reconstruction algorithms studied in the field of signal processing and that BCS-SVD has not been handpicked, we selected four different algorithms (Fig. 57): the NESTerov reconstruction Algorithm NESTA [Beck11], Gradient Projection for Sparse Reconstruction (GPSR) [Figu07], Sparsity Adaptive Matching Pursuit (SAMP) [Do08] and BCS-SVD algorithm [Akb118], we have then performed a BCS sampling of the test images in Fig. 55 using the same $\mathbf{\Phi}_{\mathrm{ECA}}^{\mathrm{T}}$ and used these algorithms to recover the images and graphic their performance in terms of PSNR over sampling subrate as we did for the different matrices of Fig. 56.

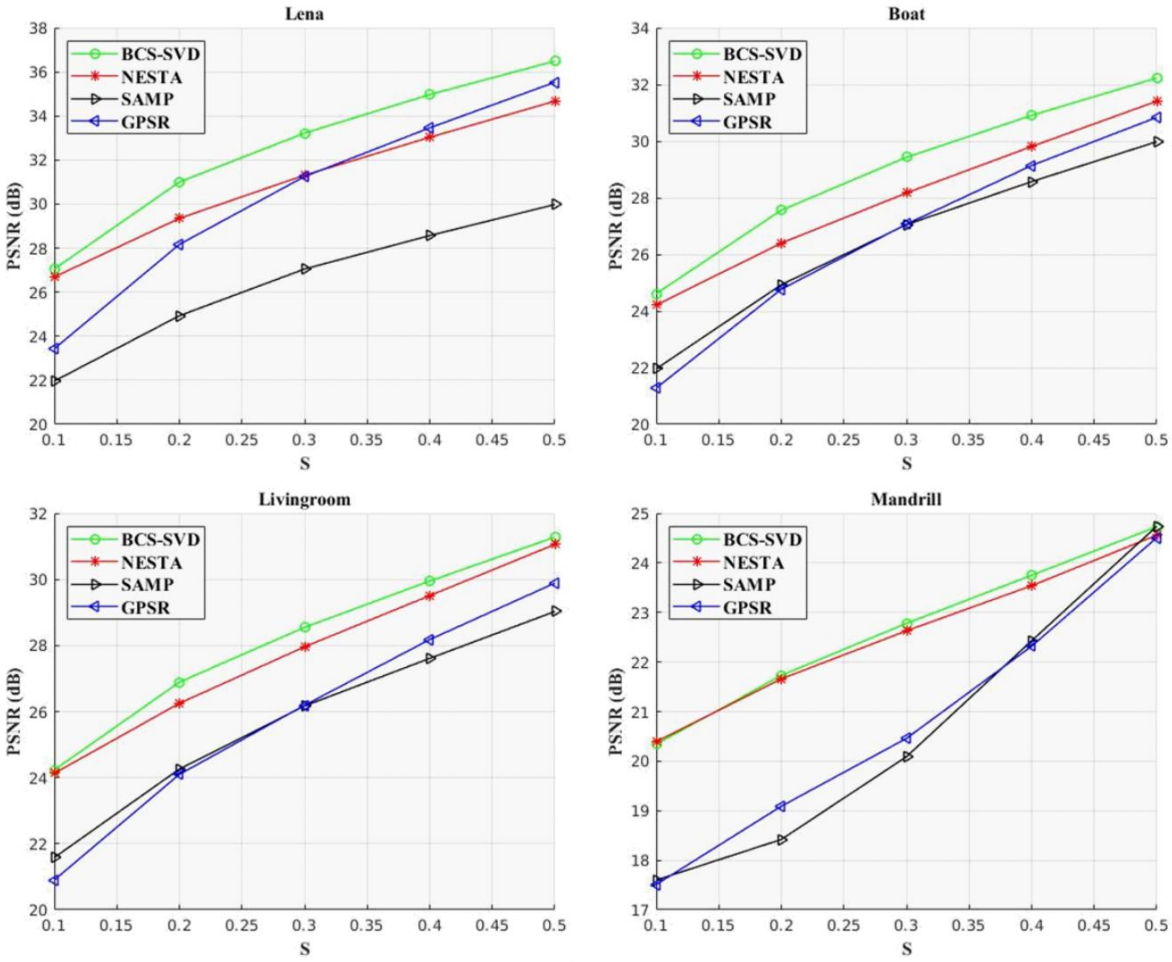

Fig. 57: PSNR of the reconstructed images using different recovery algorithms as a function of subrate (S).

During this simulation, the test images are again divided blocks of size 32×32 pixels. Fig.

57 shows the quality of reconstructed images in terms of PSNR at different subrates S. It can be observed that all this algorithms can recover the images, correctly with PSNR of the reconstructed images ranging from 23 dB to 30 dB.

CHAPTER 7
SPARSIFYING DICTIONARY BASED ON HARRIS CORNER DETECTION

Another take on Eq. (1) is that it creates the need to implement low pass filters during sampling because it is an interpolation. If we had a series of discrete points in a plane $[s(t), t]$, theoretically, could be interpolated by infinite functions of the type:

$$s(t) = \sum_i \mathrm{A}_i \sin(it) \text{ with } i = 1,2,\cdots,N \tag{62}$$

if N were to approach infinity. But, if we were to limit N to the minimum sufficient and necessary needed for $s(t)$ to cross them, then our solution would be unique. Limiting N is transposed to practice by limiting the upper end of the frequency components of the signal to be sampled before starting to take the samples $\mathbf{s}(i\Delta T)$ at intervals ΔT. But, Eq. (5) is a minimization in L_1 (i.e. optimization) or in some cases simplified to L_0 (i.e. greedy pursuit), these are not based on interpolation, they look for minima that are unique as long as the measurement matrix respects RIP Eq. (8) and the signal is k-sparse in some basis.

Saying that a measurement matrix allows a maximum number k Eq. (21) of relevant elements in a sampled signal, is like saying, in the case of standard sampling, that a converter has a sampling frequency f_s, being $f_s/2$ its Nyquist frequency.

Both the aliasing errors encountered when a sampled signal maximum frequency component is allowed to exceed $f_s/2$ and the errors in the compressed samples found when the sampled signal relevant elements exceed k translate to intrinsic errors of the generated samples. The advantage that CS has over standard sampling is that, while in standard sampling we need to implement low-pass filters before the sampling process can begin, in CS we can use sparsifying dictionaries $\mathbf{\Psi}$ a posteriori to change the basis of the signal before recovery in order to comply with sparsity requirements during reconstruction.

A dictionary forms a basis when every signal is uniquely represented as the linear

combination of the dictionary elements. As we mentioned in the introduction, CS, applied to image sampling, heavily relies on the discrete wavelet and cosine transforms, being these two domains the most suitable for image compression [Anto92]. But there are limits on how much an image can be compressed, if $\boldsymbol{\Phi}$ has a constant k that is too low, as is the case with PRNG generated binary matrices Eq. (21), then there is no basis in which reconstruction of complex images could be carried out error free. What's more, these basis have to be incoherent with the measurement matrix in order to comply with RIP of their product $\boldsymbol{\Phi\Psi}$, penalty the increase in the minimum number of compressed samples needed to achieve reconstruction [Romb09].

Since, the result on Eq. (21) makes it impossible to use recovery algorithms on samples derived from PRNG generated binary matrices without incurring in reconstruction errors we propose to use them to extract features. This operation is not to be confused with the act of extracting features from compressed samples in the compressed domain. What we are trying to do is to design a sparsifying dictionary capable of transforming the content of a set of compressed samples into information about a certain feature of the image that they represent. Such a sparsifying dictionary would not be a basis aiming to represent the whole image in a new domain; it would rather be used to prune the compressed samples and preserve only relevant information. By applying this dictionary to the samples before reconstruction, the output of a generic recovery algorithm would be the feature that we wanted to extract.

7.1 Creating a Feature Extracting Sparsifying Dictionary

The idea of creating a sparsifying dictionary capable of pruning the compressed samples and preserve only certain information sparks from the fact that, if the features to be recovered can be expressed as the value of a single coefficient of a matrix overlapping the discrete image and the number of these features approaches k, then any reconstruction algorithm should be able to recover them with minimal errors.

To demonstrate this statement we used the Harris corner detection algorithm. We chose this algorithm for its simplicity. A corner can be defined as the intersection of two edges. To determine the presence of a corner the algorithm tests each pixel considering how similar a patch centred on the pixel is to nearby largely overlapping patches. It generates a sparse matrix $\mathbf{C}$ with the same amount of elements as the pixels present in the image. This matrix has relevant values only in correspondence of the pixels where corners can be found. The matrix of parameters $\mathbf{C}$ can be obtained by computing:

$$C = Det(\widetilde{\mathbf{A}}) - h\left[Tr(\widetilde{\mathbf{A}})\right]^2 \tag{63}$$

Defining $\widetilde{\mathbf{A}}$ as:

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{X}_i^2(u,v) & \mathbf{X}_i\mathbf{X}_j(u,v) \\ \mathbf{X}_i\mathbf{X}_j(u,v) & \mathbf{X}_j^2(u,v) \end{bmatrix} \tag{64}$$

being h an empiric parameter with values in the range of $[0.04 \quad 0.15]$. The determinant and the trace of matrix $\widetilde{\mathbf{A}}$ solely depends on the partial derivatives of the image $\mathbf{X}_i$ and $\mathbf{X}_j$. First order discrete partial derivatives can be expressed as the convolution of the image $\mathbf{X}$ with the masks:

$$\mathbf{D}_i = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{D}_j = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{65}$$

Convolution of $\mathbf{X}$ by $\mathbf{D}_i$ and $\mathbf{D}_j$ can be replaced by the left multiplication of a folded version of $\mathbf{X}$ by $\mathbf{T}_{\mathbf{D}i}$ and $\mathbf{T}_{\mathbf{D}j}$, the corresponding Toeplitz matrices [Chan07] of matrices $\mathbf{D}_i$ and $\mathbf{D}_j$. Joining Eq. (63), Eq. (64) and Eq. (65), after a lengthy matrix manipulation process it would be possible to reach a form similar to:

$$\mathbf{C} = \mathbf{H}_c\mathbf{x} \tag{66}$$

being $\mathbf{H}_c$ the sparsifying dictionary that we were looking for. Given the complexity of the calculation that exceeded the purpose and field of this dissertation we have resolved to create a MATLAB experiment starting from the derivatives of the original image. This simplification helped to ease the calculations without loss of generality of the experiment results.

7.2 PERFORMANCE OF THE FEATURE EXTRACTING SPARSIFYING DICTIONARY

To analyse the benefits of using Eq. (66) to recover a set of corners instead of a whole image we devise an experiment using a 64x64 grayscale picture of Lena (Fig. 58), the first step in our experiment consisted in finding the corners of this image using the Harris corner detection algorithm. We then extracted compressed samples of the image using a PRNG generated binary measurement matrix and used the NESTA reconstruction algorithm [Beck11] to recover it. Once again we used the Harris corner detection algorithm on the recovered image. We then proceeded to use what we presented in section 7.1 and used the NESTA to recover only the corners. Lastly we compare the corner retrieved from the reconstructed image to those retrieve using our newly introduces method:

Fig. 58: (Top Left) Original Image; (Top right) Original image with Harris; (Bottom left) Nesta reconstructed image with Harris; (Bottom right) Harris Nesta corners.

We opted to use NESTA because this algorithm can be used to solve Total-Variation (TV) minimization problems [Beck11]. TV is often used to recover images from noisy and/or undersampled data. It is possible to apply this constraint to Eq. (7) and write the reconstruction problem in Eq. (9) as follows:

$$argmin\|\mathbf{C}\|_{TV} \text{ subject to } \|\mathbf{y} - \mathbf{\Phi C}\|_{l_2} < \varepsilon^2 \tag{67}$$

being $\|\mathbf{C}\|_{TV}$ equal to:

$$\|\mathbf{C}\|_{TV} = \sum_{i,j} \|\nabla \mathbf{C}[i,j]\| \tag{68}$$

To compare the performance of our sparsifying dictionary over the application of a Harris algorithm on a NESTA reconstructed image we have considered three different parameters. Distance between original corners and reconstructed corners, False Positives and False Negatives. We have repeated the process presented in Fig. 58 varying each time the number of compressed samples from a minimum of 1 to a maximum of 4096, which corresponds to the number of pixels of the original image.

Even though the graphics of false positives and false negatives in Fig. 59 are somehow cluttered, performing NESTA to recover only the corners led to an average of 11% less false negatives and 8 % less false positive.

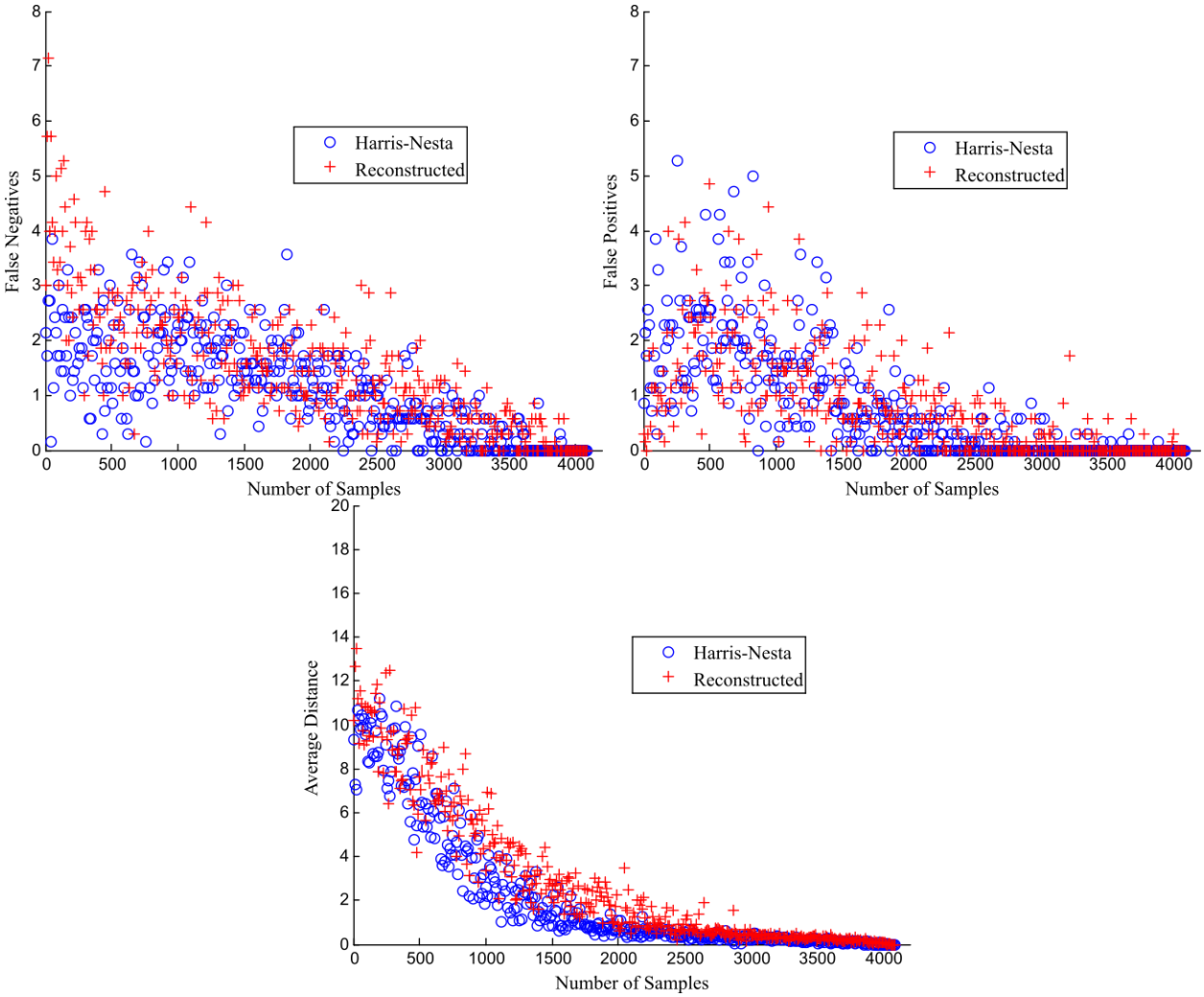

Fig. 59: (Top Left) False Negatives; (Top right) False Positives; (Bottom) Average Distance between the corners of the original image and those of the reconstructed image.

One important result presented in Fig. 59 is that the corners extracted, applying our dictionary on the samples, are closer to its original location especially when the compressed samples were low in number. This is a particularly good result because remember, given Eq. (7), the closer the number of samples gets to that of the pixels, the less is the compression on the image and to the limit when it equals the number of pixels and the problem presented in Eq. (9) is no longer underdetermined.

CHAPTER 8
CONCLUSIONS

This thesis is a work that tries to connect the theory of CS mostly studied in the field of signal processing and computer science with the implementations and practical uses of CS mainly developed in the field of microelectronic and sensor design.

This connection is created through a series of mathematical tools such as a prove of RIP (section 2.2) to study the performance of measurement matrices generated using PRNG; the PSD analysis (section 2.3) that ties the dynamic behaviour of PRNG to the mutual coherence of measurement matrices; or, the Density analysis (section 2.5) that links the output of PRNG to the Bernoulli probability distribution of true random binary measurement matrices (reminding that it is impossible to design an algorithm that implements true random number generation since that would be a contradiction in terms).

While we were adapting these tools, which are usually employed to study chaotic behaviours of complex systems [Boei16], data handling [Sara12] or other signal processing problems [Mazz08], to the analysis of measurement matrices, we mainly focused on the comparison between the two types of PRNG that are most widespread in sensor design, which are LFSR and ECA. But, the inclusive nature and rigor of these analyses could very well turn them into guidelines to study other types of one dimensional pseudo random generators that might possibly impose themselves as superior choices for the design of CS-CSI.

We also tried to attack one of the most overlooked problems in CS at least in the field of microelectronics: the enormous dynamic range needed to describe the compressed samples which makes standard ADC almost impossible. To that end we have studied two possible solutions to increase the number of bits of the output delivered by a CS-CIS. These solutions are both based on pixel output modulation, namely PFM (section 3.2.1) and PWM (section 3.2.2).

Furthermore, we have developed a CS-CIS prototype, the ‘CSImager1’ that, despite a design flaw that truncated our experiments halfway, made it possible to test and confirm the feasibility of the theory above presented.

We were able to design, develop and analyse a pulse width modulated pixel in CMOS 0.18μm 1P6M technology that, according to its response curve to varying light conditions (Fig. 49), is capable of delivering 1024 compressed samples at 30fps. This pixel can be used in a CS-CIS with an array of size 64×64 that, by means of PWM and the use of sample & accumulate elements is capable of generating 20-bit compressed samples. This achievement places within range the possibility of creating block based CS-CIS with larger block sizes that should increase the quality of the sampled images.

We have derived a new method of background subtraction from compressed samples without resorting to recursive reconstruction. This has resulted in a lightweight fast detection algorithm capable of analysing a stream of compressed samples searching for the presence of moving objects. Even if, when compared to other methods, it delivered poorer results, its speed makes it possible to process compressively sampled video streams in real time.

Thanks to our work developed in cooperation with the École d'Ingénieurs du Numérique (ISEP), we have studied the possibility of expanding the use of PRNG to design pseudo random ternary measurement matrices with minimal costs in terms of on-chip and pixel fill factor. These matrices have shown great promise when compared with the theoretical optimum, a normalised Gaussian measurement matrix and, more importantly, they clearly outperform their binary counterparts (Fig. 56)

Lastly, we have presented a new application of CS related to feature extraction. We successfully used CS reconstruction algorithms as a basis to recover Harris corners from a compressed samples simplified thorough a sparsifying dictionary Eq. (66) capable of pruning them and preserve only relevant information about the corners. This dictionary exploits the concepts of RIP and sparseness to provide an error free recovery of said corners that only depends on the measurement matrix used to obtain the samples.

REFERENCES

[Ahme74] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform". *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90 -93. Jan. 1974.

[Akan92] A. N. Akansu, R.A. Haddad. “Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets”. *San Diego: Academic Press*. ISBN 978-0-12-047141-6, 1992

[Akb118] A. Akbari and M. Trocan, "Robust Image Reconstruction for Block-Based Compressed Sensing Using a Binary Measurement Matrix". *25th IEEE International Conference on Image Processing (ICIP)*, pp. 1832-1836. Athens, Oct. 2018.

[Akb218] A. Akbari, M. Trevisi and M. Trocan, "Adaptive Compressed Sensing Image Reconstruction Using Binary Measurement Matrices". *25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 659-660. Bordeaux, Dec. 2018.

[Anto92] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies. “Image Coding Using Wavelet Transform”. *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 205 - 220. Apr. 1992.

[Arce14] G. R. Arce, D. J. Brady, L. Carin, H. Arguello, and D. S. Kittle, “Compressive coded aperture spectral imaging: An introduction,” *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 105–115, 2014.

[Bara07] R. G. Baraniuk. Compressive Sensing [Lecture Notes]. *IEEE Signal Processing Magazine*, Vol. 24, no. 4, pp. 118–121. 2007.

[Bara10] R. G. Baraniuk, V. Cevher, M. B. Wakin. “Low-Dimensional Models for Dimensionality Reduction and Signal Recovery: A Geometric Perspective”. *Proc. of the IEEE*, Vol. 98, No. 6, pp. 959-971. Jun 2010.

[Beck11] A. Becker, J. Bobin, and W. J. Candès, “NESTA: A fast and accurate first-order method for sparse recovery”. *SIAM Journal on Imaging Sciences*, Vol. 4, no. 1, pp. 1-39, 2011

[Boei16] G. Boeing. “Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction”. *Systems*, Vol. 4, No. 37. Nov. 2016.

[Bouf08] P. T. Boufounos and R. G. Baraniuk, “1-Bit compressive sensing,” *CISS 2008, 42nd Annu. Conf. Inf. Sci. Syst.*, pp. 16–21, 2008.

[Cand06] E. J. Candès. “Compressive sampling”. *Int. Congress of Mathematics*, pp. 1433-1452. Madrid, Aug. 2006.

[Cand06] E. J. Candès, J. K. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[Chan07] S. Chandrasekaran, M. Gu, X. Sun, J. Xia and J. Zhu, “A Superfast Algorithm for Toeplitz Systems of Linear Equations”. *SIAM J. Matrix Analysis Applications*, Vol. 29, pp. 1247-1266. 2007.

[Chen05] H. Chen, M. S. Asif, A. C. Sankaranarayanan, and A. Veeraraghavan, “FPA-CS: Focal plane array-based compressive imaging in short-wave infrared,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 2358–2366, 2015.

[Chen08] Y. Chen, F. Yuan and G. Khan. “A new wide dynamic range CMOS pulse-frequency-modulation digital image sensor with in-pixel variable reference voltage”. *51st Midwest Symposium on Circuits and Systems*, pp. 129-132. Knoxville, Sep. 2008.

[Chen11] D. G. Chen, D. Matolin, A. Bermak, C. Posch, “Pulse-Modulation Imaging—Review and Performance Analysis”. *IEEE Trans. on Biomedical CAS*, Vol. 5, No. 1, pp. 64-82. Feb. 2011.

[Culu03] E. Culurciello, R. Etienne-Cummings, and K. Boahen. "A Biomorphic digital image sensor," *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 2, pp. 281-294. Feb. 2003.

[Dadk15] M. Dadkhah, M. Jamal Deen, S. Shirani. “CMOS Image Sensor With Area-Efficient Block-Based Compressive Sensing”. *IEEE Sensor Journal*, Vol. 15 No. 7,

pp 3699-3710. Jul. 2015.

[Dave07] Davenport, M. Duarte, M. Wakin, J. Laskar, D. Takhar, K. Kelly, R. Baraniuk. "The Smashed Filter for Compressive Classification and Target Recognition". *Proc. SPIE Computational Imaging V*. San Jose, Jan. 2007.

[DeVo07] R. A. DeVore. "Deterministic constructions of compressed sensing matrices". *J. Complexity*, Vol. 23, pp. 918–925. 2007.

[Do08] T. T. Do, L. Gan, N. Nguyen, and T. D. Tran. "Sparsity adaptive matching pursuit algorithm for practical compressed sensing". in *Proceedings of the 42th Asilomar Conference on Signals, Systems, and Computers*, pp. 581–587. Pacific Grove, Oct. 2008.

[Ferg06] R. Fergus, A. Torralba, and W. T. Freeman. "Random Lens Imaging". *Computer Science and Artificial Intelligence Laboratory, Technical Report*. Massachusetts Institute of Technology, Sep. 2006.

[Fern16] J. Fernández-Berni, M. Suárez, R. Carmona-Galan, V Brea, R. Del Río, D. Cabello and A. Rodríguez-Vázquez. "Image feature extraction acceleration". in *Image Feature Detectors and Descriptors*, Cham, Switzerland:Springer, pp. 109-132. 2016.

[Figu07] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. "Gradient projection for sparse reconstruction: Application compressed sensing and other inverse problems". *IEEE Journal on Selected Areas in Communications*, Vol. 1, no. 4, pp. 586–597. 2007.

[Font13] R. Fontaine. "The Evolution of Pixel Structures for Consumer-Grade Image Sensors". in *IEEE Transactions on Semiconductor Manufacturing*, Vol. 26, No. 1, pp. 11-16. Feb. 2013.

[Foss97] E. R. Fossum. "CMOS image sensors: electronic camera-on-a-chip". in *IEEE Transactions on Electron Devices*, Vol. 44, No. 10, pp. 1689-1698. Oct. 1997.

[Gan07] L. Gan. "Block Compressed Sensing of Natural Images". *15th International Conference on Digital Signal Processing*, pp. 403-406. Singapore, Jul. 2007.

[Gonz92] R.C. González and R.E. Woods. "Digital Image Processing". Prentice Hall, 1992.

[Guic16] W. Guicquero, A. Dupret, and P. Vandergheynst, "An Algorithm Architecture Co-Design for CMOS Compressive High Dynamic Range Imaging," *IEEE Trans.*

Comput. Imaging, vol. 2, no. 3, pp. 190–203, 2016

[Günt10] C. S. Güntürk, M. Lammers, A. Powell, R. Saab and Ö. Yilmaz. "Sigma delta quantization for compressed sensing". *44th Annual Conference on Information Sciences and Systems (CISS)*, pp.1-6. Princeton, 2010

[Indy08] P. Indyk, "Explicit constructions for compressed sensing matrices". in *Proc. 19th Annu. ACM-SIAM Symp. Discr. Algorithms*, pp. 30–33. Jan. 2008.

[Jacq09] L. Jacques, P. Vandergheynst, A. Bibet, V. Majidzadeh, A. Schmid, and Y. Leblebici. "CMOS compressed imaging by random convolution". in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, pp. 1113–1116. Apr. 2009.

[Jacq11] L. Jacques, D. K. Hammond, and J. M. Fadili, "Dequantizing compressed sensing: When oversampling and non-Gaussian constraints combine," *IEEE Trans. Inf. Theory*, vol. 57, no. 1, pp. 559–571, 2011.

[Jafa12] S. Jafarpour, M. F. Duarte, R. Calderbank. "Beyond worst-case reconstruction in deterministic compressed sensing". *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 1852– 1856. Cambridge (USA), Jul. 2012.

[Jen90] E. Jen. "Aperiodicity in one-dimensional cellular automata". *Physica D: Nonlinear Phenomena*, Vol. 45 No. 1-3, pp 3-18. Sep. 1990.

[Jian12] H. Jiang, W. Deng, and Z. Shen. "Surveillance video processing using compressive sensing". *Inverse Problems and Imaging*, Vol. 6, no. 2, pp. 201–214. 2012.

[Kali14] B. Kaliannan, V S. Rao Pasupureddi. "A Low Power CMOS Imager Based on Distributed Compressed Sensing". *27th International Conference on VLSI Design and 13th International Conference on Embedded Systems*, pp. 534 – 538. Mumbai, Jan. 2014.

[Kati13] N. Katic, M. H. Kamal, M. Kilic, A. Schmid, P. Vandergheynst, and Y. Leblebici. "Power-efficient CMOS image acquisition system based on compressive sampling". in *Proc. IEEE Intl. Midwest Symp. Circuits Systems*, pp. 1367–1370. Aug. 2013

[Kitc05] A. Kitchen, A. Bermak, and A. Bouzerdoum. "A digital pixel sensor array with programmable dynamic range". *IEEE Tranactions. Electron Devices*, Vol. 52, No. 12, pp. 2891-22600, Dec. 2005.

[Klei01] S. Kleinfelder, S. Lim, and A. El Gamal. "A 10000 frames/s CMOS digital pixel sensor." *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 12, pp. 2049-2059, Dec. 2001.

[Kyoc19] "85 PIN GRID ARRAY PACKAGE", Kyocera Corporation, Kyoto, Japan. https://europractice-ic.com/wp-content/uploads/2019/06/CD_PGA84.pdf

[Lee18] H. Lee, D. Seo, W. Kim and B. Lee. "A Compressive Sensing-Based CMOS Image Sensor With Second-Order Σ/Δ ADCs," in *IEEE Sensors Journal*, Vol. 18, no. 6, pp. 2404-2410. 15 Mar. 2018.

[Leit18] S. Leitner, H. Wang and S. Tragoudas. "Design of Scalable Hardware-Efficient Compressive Sensing Image Sensors," in *IEEE Sensors Journal*, Vol. 18, no. 2, pp. 641-651. Jan. 2018.

[Leñe18] J. A. Leñero-Bardallo, A. Rodríguez-Vázquez. "ADCs for Image Sensors: Review and Performance Analysis", in *Analog Electronics for Radiation Detection*, Ed. Boca Raton: CRC press. Apr. 2018.

[Liut14] A. Liutkus et al., "Imaging with nature: Compressive imaging using a multiply scattering medium," *Sci. Rep.*, vol. 4, pp. 1–7, 2014.

[Lu13] W. Lu, W. Li, K. Kpalma, J. Ronsin. "Near-optimal Binary Compressed Sensing Matrix". *Manuscript submitted to IEEE Transaction on Information Theory*, Mar. 2013.

[Maji10] V. Majidzadeh, L. Jacques, A. Schmid, P. Vandergheynst and Y. Leblebici. "A (256x256) Pixel 76.7mW CMOS Imager/Compressor Based on Real-Time In-Pixel Compressive Sensing". *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2956–2959. Paris, May 2010.

[Malo11] F. Maloberti. "Data Converters", in *Analog Electronics for Radiation Detection*, Ed. Springer, New York, 2011.

[Mazz08] G. Mazzini, R. Rovatti, G. Setti. "Randomly-Flipped Linear Feedback Shift Registers: Spectral Analysis and Conjectures". *Proceedings of 2008 IEEE International Symposium on Spread Spectrum Techniques and Applications*, pp. 751–755. Bologna, Aug. 2008.

[Moch15] Futa Mochizuki, Keniichiro Kagawa, Shin-ichiro Okihara, Min-Woong Seo, Bo Zhang, Taishi Takasawa, Keita Yasutomy, Shoji Kawahito. "Single-sShot 200Mfps

5x3-Aperture Compressive CMOS image". *2015 IEEE International Conference on solid-state Circuits*. San Francisco, Feb. 2015

[Mun09] S. Mun and J. E. Fowler. "Block compressed sensing of images using directional transforms", *Proceedings of IEEE International Conference on Image Processing*, pp. 3021–3024. Cairo, Nov. 2009.

[Nage09] P. Nagesh and B. Li. "A Compressive Sensing Approach for Expression-Invariant Face Recognition". *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1518-1525, Jun. 2009.

[Ndaw15] B. A. Ndaw, D. Sow and M. Sanghare. "Construction of Maximum Period Linear Feedback Shift Registers (LFSR) (Primitive Polynomials and Linear Recurring Relations)". *British Journal of Mathematics & Computer Science*, pp. 403-406. Vol. 11, No. 4, pp. 1–24. 2015.

[Nina08] S. Ninagawa. "Power Spectral Analysis of Elementary Cellular Automata". *Complex Systems Publications, Inc.*, Vol. 17, No. 4, pp. 399–411. Dec. 2008.

[Nish11] Haridas, Nisha & Devi, Nirmala. "Efficient Linear Feedback Shift Register design for Pseudo Exhaustive Test Generation in BIST. 1". 2011. 10.1109/ICECTECH.2011.5941621.

[Ober17] R. Obermeier, J.A: Martínez-Lorenzo. "Sensing Matrix Design via Mutual Coherence Minimization for Electromagnetic Compressive Imaging Applications". *IEEE Transactions on Computational Imaging*, Vol. 1 No. 2, pp 217-229. Jun 2017.

[Oike13] Y. Oike, A. El Gamal. "CMOS Image Sensor With Per-Column S? ADC and Programmable Compressed Sensing". *IEEE Journal of Solid-State Circuits*, Vol. 48, No. 1, pp. 318 – 328. Jan. 2013.

[Pete62] D. P. Petersen and D. Middleton. "Sampling and Reconstruction of Wave-Number-Limited Functions in N-Dimensional Euclidean Spaces". *Information and Control*, Vol. 5, pp. 279–323, 1962.

[Romb09] J. K. Romberg. "Compressive sensing by random convolution". *SIAM J. Imag. Sci.*, Vol. 2, no. 4, pp. 1098–1128. Dec. 2009

[Roum08] F. Roummel, M. M. Willett, R. M. Willett. "Compressive Coded Aperture Super-resolution Image Recontruction". *IEEE International Conference on Acoustics,*

Speech and Signal Processing (ICASSP), pp. 833 – 836. Mar. 2008.

[Sara12] S. Saravanan, M. Lavanya, R. V. Sai, R. Kumar. “Design and Analysis of Linear Feedback Shift Register Based on Various Tap Connections”. *Elsevier, Procedia Engineering*, Vol. 38, pp. 640-646. 2012.

[Seit00] P. Seitz. “Solid-State Image Sensing”. *Handbook of Computer Vision and Applications*, Vol. 1, pp. 165-222, Academic Press. 2000.

[Sha012] W. Shao, H. Deng, Z. Wei. "Nonconvex Compressed Sampling of Natural Images and Applications to Compressed MR Imaging", *International Scholarly Research Notices*, Vol. 2012, (Article ID 982792). 2012.

[Shi09] G. Shi, D. Gao, D. Liu and L. Wang. "High resoluton image reconstruction: A new imager via movable random exposure". *16th IEEE International Conference on Image Processing (ICIP)*, pp. 1177-1180. Cairo, Nov. 2009

[ShuT16] X. Shu-Tao, L. Weizhi. “Construction of ternary matrices with small coherence for compressed sensing”. *Electronic Letters*, Vol. 52, No. 6, pp. 447-448. Mar 2016.

[Sten84] R. L. Stens. “Approximation of Functions by Whittaker’s Cardinal Series. In: Walter W. (eds) General Inequalities 4”. *International Series of Numerical Mathematics / Internationale Schriftenreihe zur Numerischen Mathematik / Série Internationale D’Analyse Numérique*, Vol 71. Birkhäuser, Basel. 1984.

[Trev16] M. Trevisi, R. Carmona-Galán and Á. Rodríguez-Vázquez, "Non-recursive method for motion detection from a compressive-sampled video stream". *12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp. 1-4, Lisbon, Jun 2016.

[Trev17] M. Trevisi, R. Carmona-Galán and Á. Rodríguez Vázquez. “Compressive image sensor architecture with on-chip measurement matrix generation”. *13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp.25-28, Jul. 2017.

[Trev20] M. Trevisi, A. Akbari, M. Trocan, Á. Rodríguez-Vázquez and R. Carmona-Galán. "Compressive Imaging Using RIP-Compliant CMOS Imager Architecture and Landweber Reconstruction". in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 30, no. 2, pp. 387-399, Feb. 2020.

[Trop10] J. Tropp, S. Wright. “Computational Methods for Sparse Solution of Linear Inverse

Problems". *Proceedings of the IEEE*. 2010.

[Vers09] R. Vershynin. "On the role of sparsity in Compressed Sensing and random matrix theory". *3rd International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, (10.1109/CAMSAP.2009.5413304) pp. 189 - 192. 2010.

[Waki06] M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, R. G. Baraniuk. "An Architecture for Compressive Imaging". *IEEE International Conference on Image Processing*, pp. 1273 – 1276. Oct. 2006.

[Wang06] X. Wang, W. Wang, and R. Hornsey. "A high dynamic range CMOS image sensor with inpixel light-to-frequency conversion". IEEE Transactions on Electron Devices, Vol. 53, No. 12, pp. 2988-2992. Dec. 2006.

[Wang10] Q. Wang, G. Shi. "Super-Resolution Imager via Compressive Sensing". *IEEE 10th International Conference on Signal Processing (ICSP)*, pp. 956 – 959. Oct. 2010.

[Whit15] E. T. Whittaker. "On the Functions which are represented by the Expansions of the Interpolation-Theory". *Proceedings of the Royal Society of Edinburgh*, Vol. 35, pp. 181-194. 1915.

[Wolf83] S. Wolfram. "Statistical Mechanics of Cellular Automata". *Review of Modern Physics*, Vol. 5, No. 3, pp. 601-644. Jul. 1983.

[Wolf94] S. Wolfram. "Cellular Automata and Complexity: Collected Papers". *Reading, Mass: Addison-Wesley Pub. Co*. 1994.

[Wolf17] S. Wolfram. "Oh My Gosh, It'S Covered In Rule 30S!". [online] https://writings.stephenwolfram.com/. Available at: <https://writings.stephenwolfram.com/2017/06/oh-my-gosh-its-covered-in-rule-30s/>.

[Wota10] Y. Wotao, S. Morgan, J. Yang, and Y. Zhang. "Practical Compressive Sensing with Toeplitz and Circulant Matrices". *Proceedings of SPIE - The International Society for Optical Engineering*, (7744. 10.1117/12.863527). 2010

[Xia15] S. T. Xia, X. J. Liu, Y. Jiang and H. T. Zheng. "Deterministic Constructions of Binary Measurement Matrices from Finite Geometry". *IEEE Transactions on Signal Processing*, Vol. 63, No. 4, pp. 1017-1029. Feb. 2015.

APPENDIX A

In this appendix we report the Verilog code used to carry out the experiments described in Section 4.4.

- Test pixel and event control unit of Section 4.4.1:

```
always@(posedge clk)
begin
case (current_state)
	0:	//initialize all signals and wait a button press to continue
	begin
		R1PT_E6 <= 1'b0;
		R2PT_D8 <= 1'b0;
		REPT_F8 <= 1'b0;
		RSPT_E9 <= 1'b1;
		if (push1_J15 == 0 && push2_E1 == 0)
			begin
				wait_for_me <= wait_for_me + 1'b1;
				if (wait_for_me == 5000)
					begin
						current_state <= 1;
					end
			end
	end
	1: // delay to avoid double pressing buttons
	begin
		if (push1_J15 == 1 && push2_E1 == 1)
			begin
				wait_for_me <= 0;
				current_state <= 2;
			end
	end
	2:	// load the photodiode with the desired voltages
	begin
		RSPT_E9 <= 1'b0;	// transistor M1 (Fig. 30) activated to load the photodiode
		REPT_F8 <= 1'b0;	// event generation is disabled
		current_state <= 3;
	end
	3: // load the capacitor with Vref
	begin
		R2PT_D8 <= 1'b1;
		current_state <= 4;
	end
	4:	// cancel offset
	begin
		R1PT_E6 <= 1'b1;
		current_state <= 5;
```

```
        end
        5: // pull down the first reset signal
        begin
                R1PT_E6 <= 1'b0;
                current_state <= 6;
        end
        6: // pull down the second reset signal
        begin
                R2PT_D8 <= 1'b0;
                current_state <= 7;
        end
        7:
        begin // the event begins and the machine locks waiting for a button press
                REPT_F8 <= 1'b1;        // reading enabled
                RSPT_E9 <= 1'b1;        // M1 switched off
                if (push1_J15 == 0 && push2_E1 == 0)
                        begin
                                wait_for_me <= wait_for_me + 1'b1;
                                if (wait_for_me == 5000)
                                        begin
                                                current_state <= 1;
                                        end
                        end
        end
endcase
end
```

State 0 initialises the digital control signals of the pixel.

During state 2, flipping RSPT to logic ‘0’ enables the pixel reset (Fig. 29), the cathode of the photodiode V_{cat} charges at the desired voltage V_{pix}. While this happens the connection between the cathode and the comparator is cut off by the transmission gate controlled with REPT (read signal in Fig. 32).

In state 3, by enabling the transmission gate controlled by signal R2PT the capacitor in the autozeroing comparator is loaded with the reference voltage (V_{ref}). While the capacitor is reaching the desired reference.

States 4 and 5 are used to activate the transmission gate used to remove the inverter offset. It is important that the activation of R1PT be contained in that of R2PT as shown in Fig. 33.

State 1 and 7 are idle state where the machine waits for the user input (push buttons of the FPGA board) before repeating the commands present in the other states.

- Seed register of the ECA of Section 4.4.2:

```
always@(posedge clk_R8)
begin
CA_counter <= CA_counter + 1'b1;
case (CA_counter)
        1:
        begin
                SEED_A12 <= 1'b1;
        end
        3:
        begin
                SECL_D12 <= 1'b1;
        end
        5:
        begin
                SECL_D12 <= 1'b0;
        end
        7:
        begin
                SEED_A12 <= 1'b0;
        end
        9:
        begin
                SECL_D12 <= 1'b1;
        end
        11:
        begin
                SECL_D12 <= 1'b0;
                CA_counter <= 0;
        end
endcase
end
```

States 3, 5, 9 and 11 are used to flip SECL while states 1 and 7 are used to flip SEED. The machine runs through these states by updating a 2 bits counter after every FPGA clock cycle, CA_counter that is then reset every time it reaches stage 11. This machine creates a train of pulses that is inserted into the registers. Since the output should be the same as the input just delayed, if we were probe SEEX with an oscilloscope left continuously running we should detect this same train.

During the second part of this experiment we connected $SEED_{in}$ and $SEED_{clk}$ to the two buttons of the FPGA board, the Verilog code representing to this new control system is:

```
always@(posedge clk_R8)
begin
case (current_state)
        1:      // states 1 and 2 are used to upload the seed
        begin
                if (push1_J15 == 0 && push2_E1 == 1)      // button J15 sends a logic '1' to the shift register
                begin
                        SEED_A12 <= 1'b1;
```

```
                        current_state <= 2;
                        led1_A15 <= 1'b0;
                        led2_A13 <= 1'b1;
                end
                if (push1_J15 == 1 && push2_E1 == 0)      // button E1 sends a logic '0' to the shift register
                begin
                        SEED_A12 <= 1'b0;
                        current_state <= 2;
                        led1_A15 <= 1'b1;
                        led2_A13 <= 1'b0;
                end
        end
        2: // delay to avoid double pressing buttons
        begin
                if (push1_J15 == 1 && push2_E1 == 1)
                begin
                        led8_L3 <= 1'b1;
                        wait_for_me <= wait_for_me + 1'b1;
                        if (wait_for_me == 50000000)
                        begin
                                current_state <= 3;
                        end
                end
        end
        3:      // states 3 and 4 push the seed clock forward one step
        begin
                SECL_D12 <= 1'b1;
                current_state <= 4;
                wait_for_me <= 0;
        end
        4:
        begin
                SECL_D12 <= 1'b0;
                current_state <= 1;
                led8_L3 <= 1'b1;
        end
endcase
end
```

State 1 and 2 connect the two push buttons of the FPGA board to $SEED_{in}$ so that if the first button is pushed $SEED_{in}$ will be set at logic ‘1’ and if the second button is pushed it will be set it at logic ‘0’.

Releasing either button moves the state machine to states 3 and 4 where $SEED_{clk}$ is updated one cycle.

- Sample and Accumulate of Section 4.4.3:

```
always@(posedge clk)
begin
case (current_state)
	0://initialize control signals to disable counter and array, prepare the test
	begin
		RST1_C8 <= 1'b0;
		RST2_E7 <= 1'b1;
		READ_E8 <= 1'b1; // this signal is inverted (I forgot an inverter in the design)
		CODI_D5 <= 1'b1;
		REDY_D3 <= 1'b0;
		CASC_C3 <= 1'b1; // sequentially connects the columns while disconnecting the s&a from counter
		RESA_A3 <= 1'b1;// the reset signal of the s&a is active
		led5_D1 <= 1'b1;
		led6_F3 <= 1'b1;
		led7_B1 <= 1'b0;
		led8_L3 <= 1'b0;
		if (push1_J15 == 0 && push2_E1 == 0)
		begin
			current_state <= 1;
		end
	end
	1: // delay to avoid double pressing buttons
	begin
		if (push1_J15 == 1 && push2_E1 == 1)
		begin
			wait_for_me <= wait_for_me + 1'b1;
			if (wait_for_me == 100000)
			begin
				current_state <= 2;
			end
		end
	end
	2: // this state empties the s&a registers to start the test with a clean slate
	begin
		REDY_D3 <= ~REDY_D3;
		if (REDY_D3 == 1'b0)
		begin
			Ready_counter <= Ready_counter + 1'b1;
		end
		if  (Ready_counter == 300) // when this is done the experiment starts
		begin
			CASC_C3 <= 1'b0;		// reconnect the s&a to the counter (which is at '00000001')
			RESA_A3 <= 1'b0;		// disable the reset of the s&a  so that it can accumulate
			led5_D1 <= 1'b0;
			led6_F3 <= 1'b0;
			current_state <= 3;
		end
	end
	3:
	begin
		wait_for_me <= 0;
		Ready_counter <= 0;
		REDY_D3 <= 1'b1;
		if (push1_J15 == 0 && push2_E1 == 1) // at each stroke of button 1 REDY sends a fake event
		begin
			led7_B1 <= 1'b0;
```

```
                        led8_L3 <= 1'b1;
                        current_state <= 4;
                end
                if (push1_J15 == 1 && push2_E1 == 0) // at each stroke of button 2 CASC sums all columns
                begin
                        CASC_C3 <= 1'b1;
                        current_state <= 5;
                end
        end
        4:
        begin
                REDY_D3 <= 1'b0;
                if (push1_J15 == 1 && push2_E1 == 1)
                begin
                        wait_for_me <= wait_for_me + 1'b1;
                        if (wait_for_me == 100000)
                        begin
                                led7_B1 <= 1'b1;
                                led8_L3 <= 1'b0;
                                current_state <= 3;
                        end
                end
        end
        5: // delay to avoid double pressing buttons
        begin
                if (push1_J15 == 1 && push2_E1 == 1)
                begin
                        wait_for_me <= wait_for_me + 1'b1;
                        if (wait_for_me == 100000)
                        begin
                                current_state <= 6;
                        end
                end
        end
        6:      // pressing button 1 and 2 together restarts the experiment, otherwise 'CSImager1' output stays
fixed
        begin
                if (push1_J15 == 0 && push2_E1 == 0)
                begin
                        current_state <= 0;
                end
        end
endcase
end
```

State 0 initializes the signals used in the experiment. Pressing both buttons on the FPGA board, after a short delay (state 1), the machine empties the registers of the S&A (state 2).

State 3 and 4 are used to double flip REDY when button 1 is pressed in order to generate a fake event. Since the 20 bits of the last column of the S&A are connected directly to the outputs of the prototype, this operation is reflected into a unitary increment of the chip output.

While in state 3 it is also possible to press button 2. In doing so, instead of sending an event, the 64 columns of the S&A are connected sequentially and this shifts the output of the chip upward 7 positions because it sums $64 = 2^7$ columns, all of which are loaded with the same

count (remember that the counter is fixed at ‘00000001’ so that it has added a ‘1’ for each time we pressed button 1).

State 6 is meant to restart the whole experiment upon pressing simultaneously both buttons of the FPGA board.