Thèse n° 9501

Computational Imaging SPAD Cameras

Présentée le 12 janvier 2023

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'architecture quantique
Programme doctoral en microsystèmes et microélectronique

pour l'obtention du grade de Docteur ès Sciences

par

Andrei ARDELEAN

Acceptée sur proposition du jury

Prof. D. Atienza Alonso, président du jury
Prof. E. Charbon, Dr C. Bruschini, directeurs de thèse
Prof. R. Henderson, rapporteur
Prof. M. Gupta, rapporteur
Dr G. Boero, rapporteur

École
polytechnique
fédérale
de Lausanne

2023

Examine every word you put on paper.
You'll find a surprising number that don't serve any purpose.
— William Zinsser

To Andrada…

Acknowledgements

I will start by thanking my thesis advisor, Prof. Edoardo Charbon, for giving me the opportunity to do a PhD in the AQUA laboratory. He allowed me to pursue all of my curiosities in chip design, encouraged me to take risks and motivated and supported me when times were tough. I would also like to thank my thesis co-advisor, Dr. Claudio Bruschini, for his guidance and attention to detail which ensured that my work was of high quality.

I would like to give special thanks to Arin Can Ulku with whom I worked on the same research project, co-authored a paper and shared an office for the majority of my time at AQUA. He guided me at the begining of my PhD and was always there when I had any questions about our work on FLIM. In addition, I would like to acknowledge Kazuhiro Morimoto for his collaboration on our first 3D megapixel design and for teaching me a more methodical aproach to work.

I would like to thank all my colleagues from AQUA EPFL and TUDelft, starting with Preethi Padmanabhan, Francesco Gramuglia, Scott Lindner, Samuel Burri and Myung-Jae Lee who were there at the very begining of my PhD and continuing with Augusto Ximenes, Ivan Michel Antolovic, Chao Zhang, Andrea Ruffino, Jad Benserhir, Ermanno Bernasconi, Baris Can Efe, Simone Frasca, Utku Karaca, Pouyan Keshavarzian, Ekin Kizilkan, Yang Lin, Feng Liu, Tommaso Milanese, Paul Mos, Emanuele Ripiccini, Ming-Lo Wu, Halil Kerim Yildirim, Jiuxuan Zhao, as well as all the ones who have recently joined but I haven't had the chance to collaborate with.

I would also like to mention Brigitte Khan and Begonia Tora, who have helped me with personal issues that extended far beyond their duty as administrative staff. Additionaly, I would like to thank Michael and Fio for their friendship and support.

I would also like to acknowledge my parents and grandparents for supporting me throughout my academic career and Yuki, for sitting with me during the sleepless tapeout nights and reminding me I had to go home.

Last but definitely not least, I would like to thank Andrada for always being by my side throughout our 11 years of Bachelor, Masters and Doctoral studies and without whom I would not have been here today.

Neuchâtel, November 2022 Andrei Ardelean

Abstract

Vision systems built around conventional image sensors have to read, encode and transmit large quantities of pixel information, a majority of which is redundant. As a result, new computational imaging sensor architectures were developed to preprocess the raw pixel data and reduce the amount of information that needs to be read from the sensor. With the emergence of large format single-photon avalanche diode (SPAD) imagers, the need for on-chip processing has become more evident, as the output data rate of such detectors is pushing the limit of even modern interfaces.

The aim of the thesis is to develop sensor architectures for computational imaging that overcome limitations of conventional SPAD imagers and can operate at high frame rates with manageable output data rates. Three sensors are designed in different technology nodes, from 180 nm 2D to 3D-stacked 45/22 nm and 180/16 nm, backside and front-side illuminated.

A novel token-based readout technique is developed to improve system frame rate by reducing the readout time through omission of dark pixels. The technique is implemented in *kilo*Phase, a 32 × 32 gated SPAD imager with vector processing capabilities that can achieve 4.38 ns gates and can operate at 227 fps in 10 bit intensity mode, a 12% increase compared to conventional readout. An improved, massively parallel version of the architecture is implemented in *Mega*Phase, a 1-megapixel SPAD imager consisting of 16384 processing cores that can perform addition and multiplication operations on the raw pixel data. The SPADs can be binned with multiple granularities to increase the pixel dynamic range and reduce the required exposure time. Simulations show a frame rate increase of up to 170× when operating as an intensity imager and 56× in gated fluorescence lifetime imaging (FLIM) mode.

Finally, *Ultra*Phase, the first fully reconfigurable SPAD processing architecture is developed and consists of 18 independent processors running at 140 MOPS with a power consumption of less than 94.6 GOPS/W. Each processor can execute up to 256 instructions per program and contains a reconfigurable front end that can implement a wide range of combinatorial functions and a timing module that can be configured to measure photon arrival timestamps.

Key words: computational imaging, single-photon avalanche diode (SPAD), fluorescence lifetime imaging microscopy (FLIM), token-readout, parallel processing

Résumé

Les systèmes de vision construits autour de capteurs d'images conventionnelles doivent lire, encoder et transmettre de grandes quantités d'informations de pixels, dont la majorité est redondante. Par conséquent, de nouvelles architectures de capteurs d'imagerie computationnelle ont été développées pour prétraiter les données de pixels brutes et réduire la quantité d'informations qui doivent être lues à partir du capteur. Avec l'émergence des imageurs grand format avec la diode à avalanche à photon unique (SPAD), le besoin d'un traitement sur puce est devenu essentiel, car le débit de données de sortie de ces détecteurs repousse les limites des interfaces, même modernes.

L'objectif de la thèse est de développer des architectures de capteurs pour l'imagerie numérique qui surmontent les limites des imageurs conventionnels SPAD et qui peuvent fonctionner à des fréquences d'images élevées avec des débits de données de sortie gérables. Trois capteurs sont conçus dans différents nœuds technologiques, de 180 nm 2D à 3D empilés 45/22 nm et 180/16 nm, éclairés à l'arrière et à l'avant.

Une nouvelle technique de lecture basée sur des jetons est développée pour améliorer la fréquence d'images du système en réduisant le temps de lecture par omission de pixels sombres. La technique est mise en oeuvre en *kilo*Phase, un imageur SPAD 32×32 avec des capacités de traitement vectoriel qui peut atteindre des portes de 4.38 ns et peut fonctionner à 227 fps en mode d'intensité de 10 bits, soit 12% d'augmentation par rapport à la lecture conventionnelle. Une version améliorée et massivement parallèle à l'architecture est implémentée dans *Mega*Phase, un imageur SPAD de 1 mégapixel composé de 16384 cœurs de traitement qui peuvent effectuer des opérations d'addition et de multiplication sur les données de pixels brutes. Les SPAD peuvent être regroupés avec plusieurs granularités pour augmenter la plage dynamique des pixels et réduire le temps d'exposition requis. Les simulations montrent une augmentation de la fréquence d'images jusqu'à 170× lorsqu'ils fonctionnent comme un imageur d'intensité et 56× en mode de microscopie d'imagerie à durée de vie de fluorescence (FLIM).

Enfin, *Ultra*Phase est la première architecture de traitement SPAD entièrement reconfigurable qui se compose de 18 processeurs indépendants et qui fonctionnent à 140 MOPS, avec une consommation électrique inférieure à 94.6 GOPS/W. Chaque processeur peut exécuter jusqu'à 256 instructions par programme et contient un circuit frontal reconfigurable. Ces circuits peuvent implémenter un large éventail de fonctions combinatoires. De plus, le processeur contient un module de temporisation configurable pour mesurer les horodatages d'arrivée des photons.

Mots clés : imagerie computationnelle, diode à avalanche à photon unique (SPAD), microscopie d'imagerie à durée de vie de fluorescence (FLIM), lecture de jeton, traitement parallèle

Contents

Acknowledgements **i**

Abstract (English/Français) **iii**

List of figures **xi**

List of tables **xv**

List of acronyms **xvii**

1 Introduction **1**

1.1 Computational imaging 1
1.2 FLIM 5
1.2.1 Photon counting techniques 6
1.2.2 Phasor-based FLIM 8
1.3 SPAD 11
1.3.1 Technology and implementation 11
1.3.2 Metrology 12
1.4 Research motivation 13
1.5 Scientific contributions 14
1.6 Thesis structure 15

2 *kilo*Phase **17**

2.1 Motivation 18
2.2 Processing flow 20
2.3 Architecture 22
2.3.1 Pixel 22
2.3.2 Token-passing readout 24
2.3.3 Bus manager 26
2.3.4 Accumulator bank 26
2.3.5 Computational unit 27
2.3.6 RAM bank 28

2.4 Implementation . . . 30
2.5 Characterization . . . 34
2.5.1 Camera system . . . 34
2.5.2 Breakdown voltage . . . 35
2.5.3 Dark count rate . . . 36
2.5.4 Crosstalk . . . 39
2.5.5 PDP . . . 40
2.5.6 Power consumption . . . 40
2.5.7 Gate profile . . . 41
2.5.8 Frame rate . . . 43
2.5.9 Data rate . . . 47
2.5.10 Microlenses . . . 47
2.6 Conclusion . . . 49

3 *Mega*Phase **53**
3.1 Motivation . . . 53
3.2 Processing flow . . . 55
3.3 Architecture . . . 56
3.3.1 Pixel . . . 56
3.3.2 Computational unit . . . 58
3.3.3 Readout . . . 59
3.4 Implementation . . . 62
3.4.1 Top tier . . . 62
3.4.2 Bottom tier . . . 63
3.5 Design verification . . . 68
3.6 Conclusion . . . 68

4 *Ultra*Phase **73**
4.1 Motivation . . . 73
4.2 Architecture . . . 74
4.2.1 Reconfigurable front end . . . 75
4.2.2 Timing module . . . 76
4.2.3 Processing module . . . 77
4.2.4 Control module . . . 80
4.2.5 Processor array . . . 82
4.3 Implementation . . . 84
4.3.1 Top tier . . . 84
4.3.2 Bottom tier . . . 87
4.4 Programming . . . 91
4.4.1 Instructions . . . 91

4.4.2 Assembler 94
4.4.3 Compiler 95
4.5 Characterization 96
4.5.1 Power consumption 96
4.5.2 Timing performance 97
4.6 Applications 98
4.6.1 ToF histogram compression 98
4.6.2 LSTM LiDAR 102
4.7 Conclusion 107

5 Conclusion **111**
5.1 Summary 111
5.2 Future work 113

Appendix **115**

Bibliography **119**

Chip gallery **137**

List of publications **139**

Curriculum vitae **141**

List of Figures

1.1 Analog CVIS architecture . . . 3
1.2 Digital CVIS architecture . . . 4
1.3 Bandwidths of common interfaces . . . 5
1.4 TCSPC vs. time gating. . . . 6
1.5 Phasor FLIM plot . . . 9
1.6 Phasor FLIM plot for a double-exponential decay . . . 10
1.7 SPAD operation . . . 11
1.8 Example SPAD cross section . . . 13

2.1 *kilo*Phase readout scheme . . . 18
2.2 *kilo*Phase theoretical maximum speedup . . . 19
2.3 *kilo*Phase block diagram . . . 20
2.4 *kilo*Phase pixel schematic . . . 23
2.5 *kilo*Phase pixel layout . . . 23
2.6 *kilo*Phase token-passing schematic . . . 24
2.7 *kilo*Phase token-passing principle . . . 25
2.8 *kilo*Phase Bus manager circuit schematic for one row of the array . . . 27
2.9 *kilo*Phase computational unit schematic . . . 28
2.10 *kilo*Phase RAM schematic . . . 29
2.11 *kilo*Phase gate and recharge propagation . . . 31
2.12 *kilo*Phase gate and recharge tree . . . 31
2.13 *kilo*Phase power distribution and decoupling capacitors . . . 32
2.14 *kilo*Phase CoB PCB . . . 34
2.15 *kilo*Phase system . . . 35
2.16 *kilo*Phase breakdown voltage . . . 36
2.17 *kilo*Phase cooling assembly . . . 37
2.18 Median *kilo*Phase DCR . . . 38
2.19 *kilo*Phase DCR population distribution . . . 38
2.20 Median *kilo*Phase DCR across the entire array . . . 39
2.21 Average *kilo*Phase crosstalk . . . 40
2.22 Photon detection probability . . . 41

2.23 Average *kilo*Phase power consumption 42
2.24 *kilo*Phase gate edge characteristics 44
2.25 *kilo*Phase gate skew 44
2.26 *kilo*Phase gate width range 45
2.27 *kilo*Phase intensity mode frame rate 46
2.28 *kilo*Phase microlenses 48
2.29 *kilo*Phase microlense concentration factor 49

3.1 *Mega*Phase structure 54
3.2 *Mega*Phase readout scheme 55
3.3 *Mega*Phase pixel schematic 58
3.4 *Mega*Phase pixel layout 59
3.5 *Mega*Phase computational unit block diagram 60
3.6 *Mega*Phase SPAD binning 60
3.7 *Mega*Phase quadrant readout 61
3.8 *Mega*Phase ROI readout 61
3.9 BSI 3D-stacked SPAD sensor structure 62
3.10 *Mega*Phase top tier pixel layout 63
3.11 *Mega*Phase processing core layout 64
3.12 *Mega*Phase gate distribution 65
3.13 *Mega*Phase intensity mode simulation results 70
3.14 *Mega*Phase computational mode simulation results 71

4.1 *Ultra*Phase processing core block diagram 74
4.2 *Ultra*Phase processing core reconfigurable front end 76
4.3 *Ultra*Phase processing core pixel grouping 77
4.4 *Ultra*Phase Timing module schematic 78
4.5 *Ultra*Phase Processing module schematic 79
4.6 *Ultra*Phase instruction machine code 82
4.7 *Ultra*Phase array of processing cores 83
4.8 *Ultra*Phase 3D structure 84
4.9 *Ultra*Phase top tier pixel schematic 85
4.10 *Ultra*Phase top tier pixel layout 86
4.11 *Ultra*Phase top tier micrograph 86
4.12 *Ultra*Phase processing core layout 88
4.13 *Ultra*Phase bottom tier 88
4.14 *Ultra*Phase assembly language 95
4.15 *Ultra*Phase power consumption 97
4.16 *Ultra*Phase timing module single shot measurements 98
4.17 Encoding matrices 99

4.18 LSTM cell structure 103
4.19 LSTM scheduling graph 104
4.20 Clustering of *Ultra*Phase cores for LSTM computation 106
4.21 LSTM execution time 106
4.22 *Ultra*Phase LSTM setup 108

5.1 Experimental setup for light-in-flight imaging 116
5.2 Light-in-flight measured data 117
5.3 Reconstructed 4D light-in-flight path 117

List of Tables

2.1 *kilo*Phase power consumption 41
2.2 *kilo*Phase state of the art comparison 51

3.1 *Mega*Phase frame rate and throughput 57
3.2 Library corners used for the *Mega*Phase timing analysis. 67

4.1 *Ultra*Phase Timer module configurations 78
4.2 *Ultra*Phase Byte selector functions 79
4.3 *Ultra*Phase ALU operations 80
4.4 Library corners used for the *Ultra*Phase timing analysis. 90
4.5 *Ultra*Phase instruction operands 92
4.6 *Ultra*Phase CMP instruction effect 92
4.7 *Ultra*Phase Timing module resolution 97
4.8 Histogram encoding performance 101
4.9 *Ultra*Phase LSTM implementation RAM usage 105
4.10 *Ultra*Phase state of the art comparison 109

List of acronyms

ALU	arithmetic and logic unit
APD	avalanche photodiode
BEOL	back-end-of-line
BSI	backside illuminated
CCD	charge-coupled device
CIS	CMOS image sensor
CMOS	complementary metal-oxide-semiconductor
CoB	chip on board
CPU	central processing unit
CVIS	CMOS vision sensor
DCR	dark count rate
DMUX	demultiplexer
ESD	electrostatic discharge
FEOL	front-end-of-line
FLIM	fluorescence lifetime imaging microscopy
FPGA	field programmable gate array
FSI	front-side illumination
FSM	finite state machine
G-APD	Geiger-mode avalanche photodiode
GPU	graphics processing unit
IRF	impulse response function
LiDAR	light detection and ranging
LSTM	long short-term memory
LUT	lookup table
LVS	layout versus schematic
MAC	multiply-accumulate (operation)
MUX	multiplexer
NA	numerical aperture
NMOS	n-type metal-oxide-semiconductor
PC	program counter register
PCB	printed circuit board
PDE	photon detection efficiency

PDP	photon detection probability
PLL	phase-locked loop
QE	quantum efficiency
RAM	random-access memory
ROI	region of interest
ROM	read-only memory
SNR	signal-to-noise ratio
SoC	system on a chip
SPAD	single-photon avalanche diode
TDC	time-to-digital converter
TEC	thermoelectric cooler
ToF	time-of-flight
TSV	through-silicon via
XOR	exclusive or

1 Introduction

1.1 Computational imaging

Computational imaging can be defined as the process of recovering optically encoded scene information from measured data through the use of image processing algorithms. The technique has applications in a variety of fields such as fluorescence lifetime imaging microscopy (FLIM) [1]–[3], compressive imaging [4]–[6], depth imaging [7]–[11], spectral imaging [12]–[14], light field imaging [15]–[17] and computed tomography [18]–[20], to name only a few. In addition, computational imaging techniques can overcome hardware limitations of both optics and sensors by transferring the burden to the software domain where algorithm improvements are easier to come by. Good examples of this are coded aperture imaging systems [21]–[23] or super-resolution microscopy techniques [24]–[28].

The constant evolution seen by CMOS image sensors (CIS) in the last decades has resulted in sensors of 71 megapixels, pixel pitch of 1.1 µm, data rates above 10 Gpx/s and architectures that include preprocessing electronics to enhance image quality [29]–[33].

However, this increase in speed and resolution comes at a cost when the image sensors are used in complex hardware-software vision systems where they capture images that are then delivered for processing to other blocks [34], [35]. Raw pixel data is largely redundant and reduced subsets of samples can successfully be used to extract the same information contained in the original data [36]. As a result, a vision system built around a conventional image sensor needs to read, encode, transmit and store a large quantity of data, some of which contains no information [29]. For this reason, a new type of image sensor was developed, the *computational imager* or *CMOS vision sensor* (CVIS), a device that outputs image *features* or *decisions* resulting from performing vision tasks right at the imager plane [29].

In general, CVIS can be classified into two categories [29]:

- *Specific-purpose* architectures that are dedicated to a specific task implemented in hardware, such as histogramming photon arrival times [37]–[40], kernel convolutions [41] or biomorphic architectures [42]–[45].
- *General-purpose* architectures which contain analog and digital circuits capable of storing and executing user-selectable instruction sequences [46]–[53].

Contemporary general-purpose CVIS are mixed signal designs that try to combine the best of both worlds. The analog front ends integrate well with the CMOS pixel and are a fast and low power method of implementing nonlinear functions commonly used in image processing algorithms. The digital processing blocks are flexible, robust and can be reconfigured through software. The most efficient architectures so far employ arrays of multi-functional mixed-signal processing elements that complete computationally intensive vision tasks in a fully parallel fashion for all the pixels or for one subsection at a time [29], [54].

Massively parallel vision sensors with image formats of up to 256 × 256 have already been developed in academia [47], [48]. These devices can operate at high frame rates and are very well suited for implementing simple image processing algorithms such as edge detection, thresholding and median filtering. The disadvantages come from performing these operations in the analog domain, which requires the use of a large number of biasing signals, therefore multiple digital to analog converters, and analog memories. Every operation to and from the memory contains a signal dependent error component which accumulates with every iteration and limits the maximum processing speed of the device. In addition, stored analog values degrade over time at rates of up to 2.8 % per second, which makes it necessary to switch to and from the digital domain in order to use long term memories for applications where inter-frame storage is required, such as motion detection [47].

Figure 1.1 is a block diagram of the large format CVIS architecture described in [48]. The design consists of an array of analog processing elements, one for each pixel, that can communicate with their nearest neighbors and can simultaneously execute the same instruction. As described before, computing in the analog domain requires the use of biasing, control and error correction circuitry, which can be seen in this design as well. The chip has been successfully used in complex applications such as convolutional neural networks [55], [56].

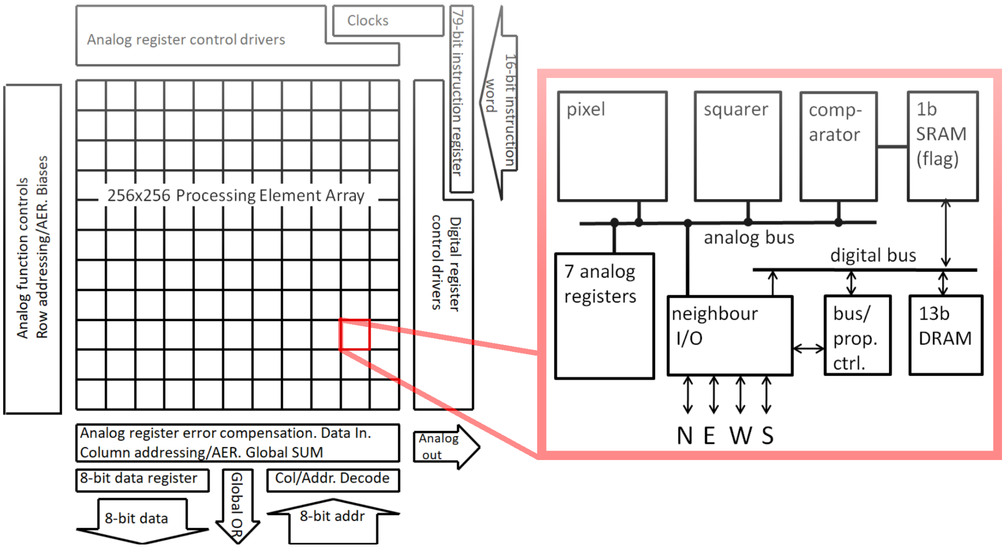


Figure 1.1: Block diagram of the large format CVIS architecture from [48]. The design consist of an array of analog processing elements that can communicate with their nearest neighbors and can simultaneously execute the same instruction. Biasing, control and error correction circuitry, typical for analog processing implementations, can be seen in the design. Figure sourced from [56].

More recent architectures, like the one shown in Figure 1.2, rely on an analog-to-digital conversion right after the pixel, followed by processing on small digital elements. These types of CVISs have already been proven capable of running more complex tasks such as gesture classification with neural networks at frame rates of more than 1000 fps and even facial recognition [49].

Commercial systems such as Teledyne Eye-RIS [46] are currently available for industrial applications in machine vision, automotive and security. Achieving 10 kfps while processing 176×144 resolution images is now possible and can replace complex vision systems with a compact and low power alternative.

The CVIS systems previously described were all based on analog pixels consisting of photodiodes which operate in the linear regime, however, a completely digital pixel built around single-photon avalanche diodes (SPADs) is arguably better suited for processing architectures. Initially requiring custom processes, SPADs can now be fabricated using standard CMOS, allowing the integration of readout circuitry in the pixel [57]–[62]. SPAD imager architectures are now available with in-pixel digital counters, active quenching and recharging circuits and even time-to-digital converters (TDC) [39], [63]–[65]. Developments in deep submicron technology and 3D-stacked processes have increased pixel complexity even further and the spatial resolution of

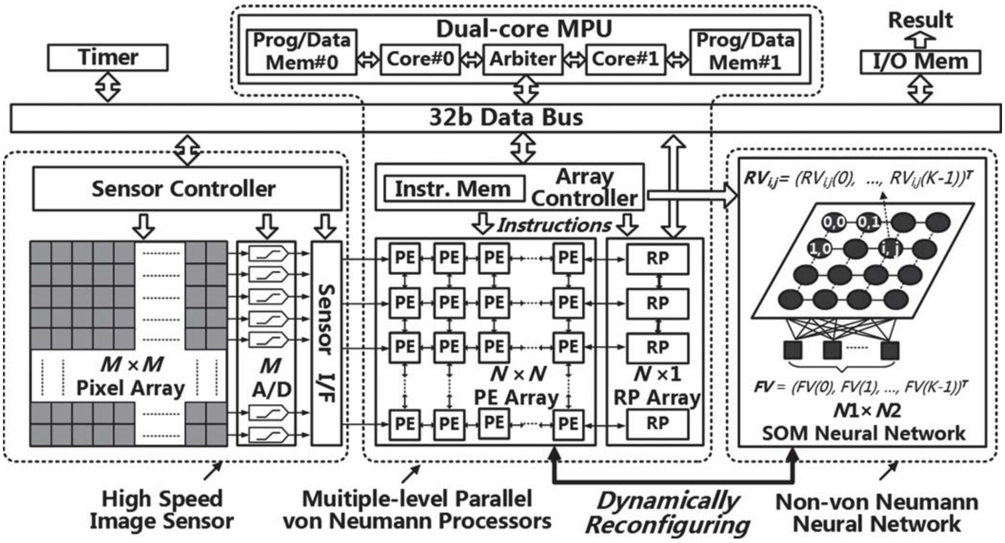


Figure 1.2: Block diagram of the large format CVIS architecture from [49]. The design relies on immediate analog-to-digital conversion of the pixel value which is then processed by an array of small digital elements. In this case, an additional dual-core CPU is needed to manage the chip. Figure sourced from [49].

SPAD imagers has reached and surpassed the megapixel level [66]–[68]. More details about the SPAD structure and metrology are given in Chapter 1.3.

The emergence of large format SPAD imagers has made the significant advantage of the reduced output data rates of computational imagers even more relevant. The amount of raw data that large format SPAD imagers generate is excessive, as state of the art pixels can operate at hundreds of Mcps [69], [70], resulting in data rates in the order of tens of GB/s, a challenge even for modern data interfaces.

Figure 1.3 shows a comparison between the data rates of common interfaces and the data rate of a 512 × 512 SPAD imager published in 2019 operating at full speed [71]. Even in this case, convenient interfaces such as USB3.0 and PCIe are either insufficient or require multiple channels. The interfaces that can natively handle this data rate are either newly standardized and not yet available or extremely demanding on hardware, requiring for example high speed grade FPGAs and storage solutions. There is a clear conclusion that in order to take full advantage of the high speed single-photon capabilities of these types of detectors, data processing needs to take place on chip, to reduce data rates and, among others, the associated power consumption.

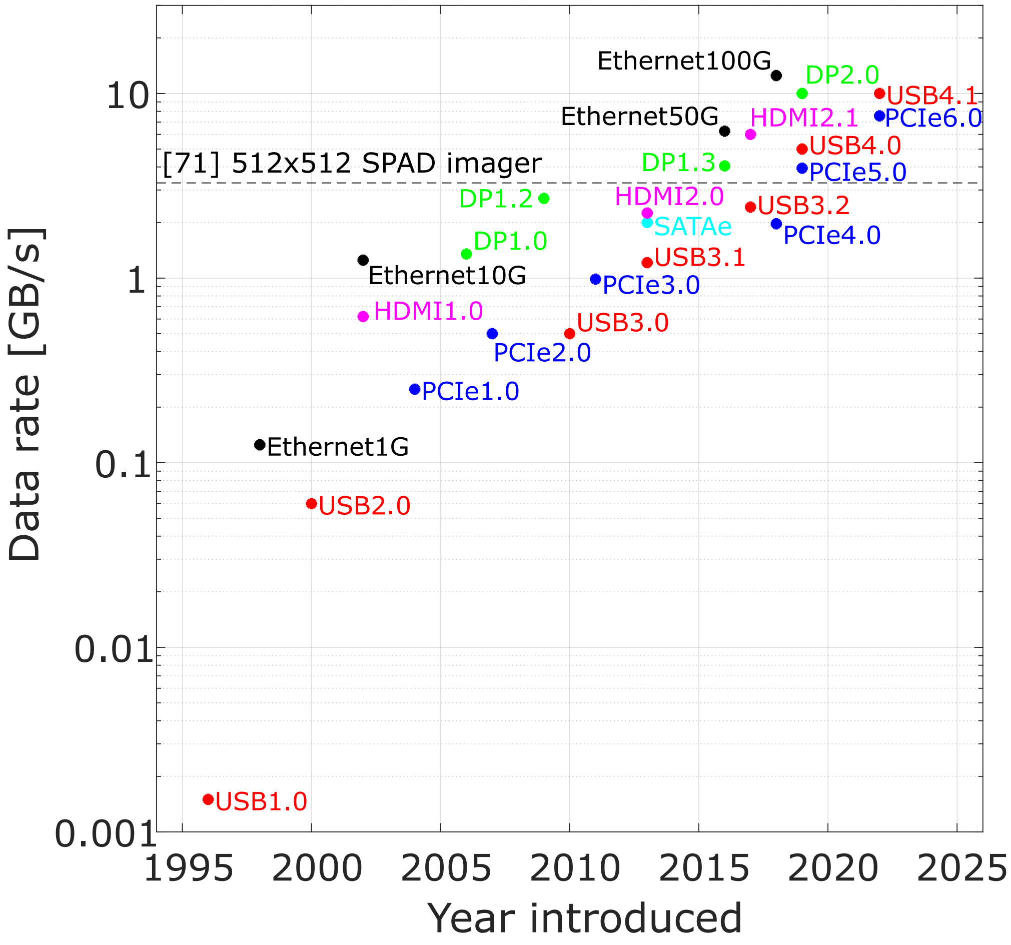
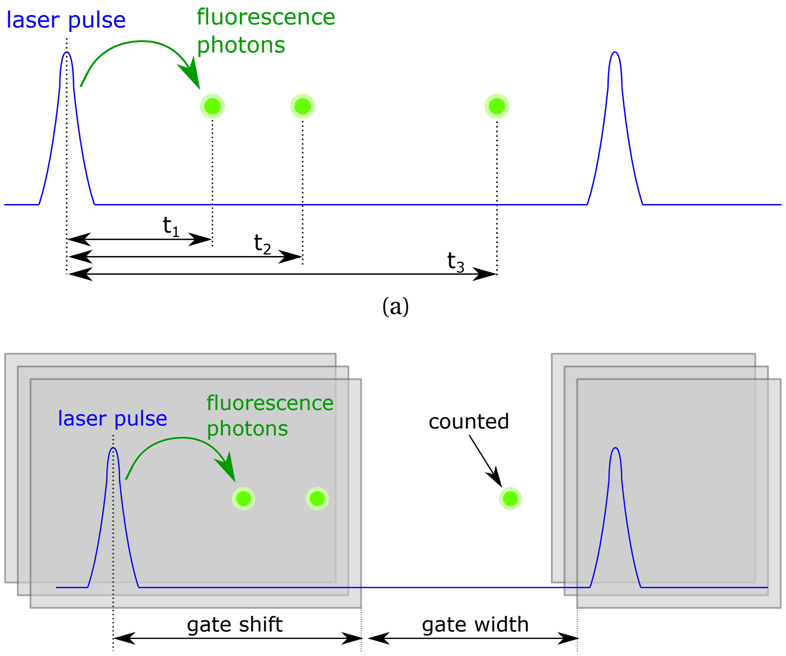


Figure 1.3: Bandwidths of common interfaces compared with the output data rate of [71], a state of the art large format SPAD imager published in 2019 operating at full speed.

1.2 FLIM

Fluorescence lifetime imaging microscopy (FLIM) is a computational imaging technique based on the differences in timing response of fluorescent molecules. The average delay between the excitation and emission of a fluorophore follows an exponential distribution characterized by a specific decay rate or *lifetime* that is dependent on the biochemical environment but immune to signal intensity variations. As a result, high contrast images with selectively labelled features can be formed even when background light or photon scattering effects are present. The dependency on the environment can be used to determine pH levels, viscosity, oxygen and ion concentrations, as well as conformal changes of molecules and interactions between them at distances below the diffraction limit [72]–[80].

1.2.1 Photon counting techniques



(a)

(b)

Figure 1.4: (a) In TCSPC the three distinct photon arrival times t_1, t_2 and t_3 are recorded w.r.t. the laser pulse. (b) In time gating, only the photon that arrives inside the gate window is registered and assigned to the time bin corresponding to the current gate shift.

TCSPC

Time-correlated single-photon counting (TCSPC) is a technique that measures the difference in arrival times between the excitation pulse and the photons emitted by the sample (see Figure 1.4a). The process is repeated multiple times until the fluorescence decay profile is obtained. Due to the high signal-to-noise ratio (SNR), TCSPC is the primary reference for many time-resolved measurements, from FLIM to light detection and ranging (LiDAR) and optical tomography [81].

TCSPC performance is mainly affected by two factors: dead time and timing resolution; characteristics of both the photodetector and the electronics. Early TCSPC hardware was limited to single channel systems that suffered from performance degradation when extended to multi channel versions. However, the invention of laser scanning confocal microscopes lead to TCSPC-based FLIM becoming widely used at the start of the XXI-st century [82].

State-of-the-art commercial TCSPC systems can achieve picosecond timing resolutions and sub nanosecond dead times, however, these devices are bulky pieces of equipment that can handle only up to 64 channels [83]. Recently, widefield TCSPC architectures have emerged based on position-sensitive detectors, most commonly microchannel plates (MCP) coupled with charge-coupled devices (CCD) or CMOS cameras or arrays of native single-photon detectors such as SPADs or superconducting nanowires. Unfortunately, MCP-based systems are limited to MHz frame rates, while SPAD arrays have low fill factor and a limited number of timing channels.[37]

Time gating

Time gating consists of acquiring timing information using square-shaped sampling windows created in the readout electronics that are phase-shifted with respect to the excitation pulse (see Figure 1.4b). Initially, heterodyning was used to implement the required phase shifts and modulation of the photodetector gain was required, but state of the art electronics can now achieve picosecond shifts with ease through the use of high frequency phase-locked loops (PLL) or delay lines while keeping the photodetector bias constant [84].

In addition, multiple sampling windows can be generated simultaneously, achieving 100 % duty cycle w.r.t. the excitation period and as a result, high count rates. Conversely, the temporal resolution is inferior to TCSPC because the latter can achieve narrower time bins. In addition, narrower gates require more parallel channels in order to maintain the same duty cycle, which leads to fill factor degradation and substantial data rate increases.

Time-resolved CCD and CMOS cameras implement digital time gates as multitap pixels and use external voltage modulation to direct photoelectrons to the tap corresponding to the current sampling window. On the other hand, SPAD imagers use gating transistors between the photodiode and the memory cell.

The difficulties of implementing in-pixel time tagging electronics without degrading the fill factor, coupled with the count rate penalties associated with sharing these circuits among multiple pixels has caused time-gating to become the main method of implementing large format time-resolved SPAD imagers [66], [67], [71] and for this reason was implemented in all the architectures presented in this thesis.

1.2.2 Phasor-based FLIM

Both previously mentioned photon counting techniques are capable of reaching video-rate speeds while detecting sufficient photons to estimate the fluorophore lifetime with acceptable precision. The bottleneck has historically been the data processing step that employs slow and resource intensive lifetime estimation methods. Initially, exponential curves were fitted to the measured photon arrival time histograms using least squares or maximum likelihood estimation methods. [85], [86] These iterative tuning methods are computationally complex and perform poorly in terms of speed when handling multi-exponential decays. In addition, they require a priori information about the exponential models, which in some cases is undesirable.

Various methods of improving lifetime estimation speeds have been developed, starting from the use of graphics processing units (GPU) to parallelize algorithm execution in detriment of power consumption and cost [87], to the use of less complex estimation algorithms such as Laguerre expansion [88], rapid lifetime determination [89]–[91] and the center of mass method [92]–[94]. Unfortunately, the less complex algorithms suffer from reduced lifetime precision or require narrow time bins.

Phasor-based FLIM has become a popular method for fast lifetime analysis because it only involves elementary arithmetic operations, doesn't require a priori information and offers a graphical representation of all the lifetimes from the sample [95]. The method is commercially available in data analysis systems such as [96] used by [97], [98]. The method consists of using the measured photon count I_k of each time bin with associated delay t_k to compute the phasor given by:

$$z = g + js = \frac{\sum_{k=1}^{N_{bins}} I_k[cos(2\pi f t_k) + jsin(2\pi f t_k)]}{\sum_{k=1}^{N_{bins}} I_k}, \tag{1.1}$$

where N_{bins} is the total number of bins/gates and f is the phasor frequency, typically chosen to be equal to the laser repetition frequency. When plotted on the complex plane, all phasors z corresponding to a single exponential decay are located in the first quadrant on a semicircle centered at (0.5, 0), with long lifetimes closer to the origin (see Figure 1.5). The lifetime of a single-exponential decay can be determined as:

$$\tau = \frac{1}{2\pi f}\frac{s}{g}. \tag{1.2}$$

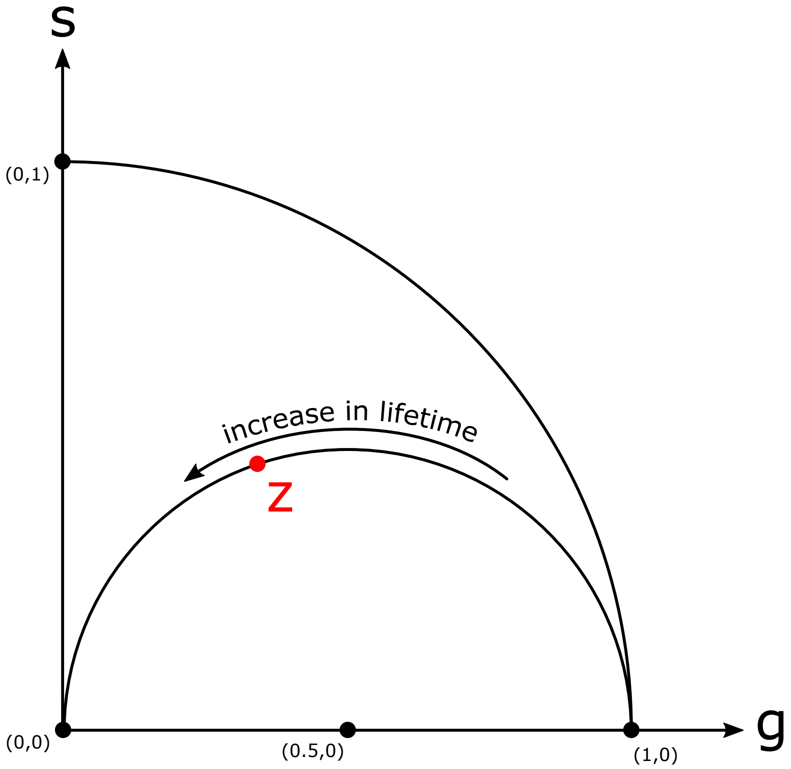


Figure 1.5: For a single-exponential decay, phasor z is located on the semicircle centered at (0.5,0), with lifetimes increasing counterclockwise.

In the case of multi-exponential decays, the phasors are located inside the semicircle, between the phasors of the single-exponential components. In the simplest case, a double-exponential, the phasor is located on the chord formed by the phasors of the two components (see Figure 1.6), splitting it into two segments of length d_1 and d_2, the ratio of which can be used to determine the ratio of the two components:

$$r_1 = \frac{d_1}{d_1 + d_2}, \tag{1.3}$$

where r_1 is the phasor ratio of the first component. This technique was used in [99] to characterize various mixtures of Rhodamine6G and Cy3B.

It is easy to observe that the implementation of phasor-based FLIM is straightforward, the most complex operation being the division. The sine and cosine terms can be computed a priori and treated as constants because t_k and f are known. The numerator and denominator from (1.1) can be computed concurrently with the acquisition and the normalization can be skipped entirely if only τ is of interest.

Another important advantage of this method is the ease with which the impulse

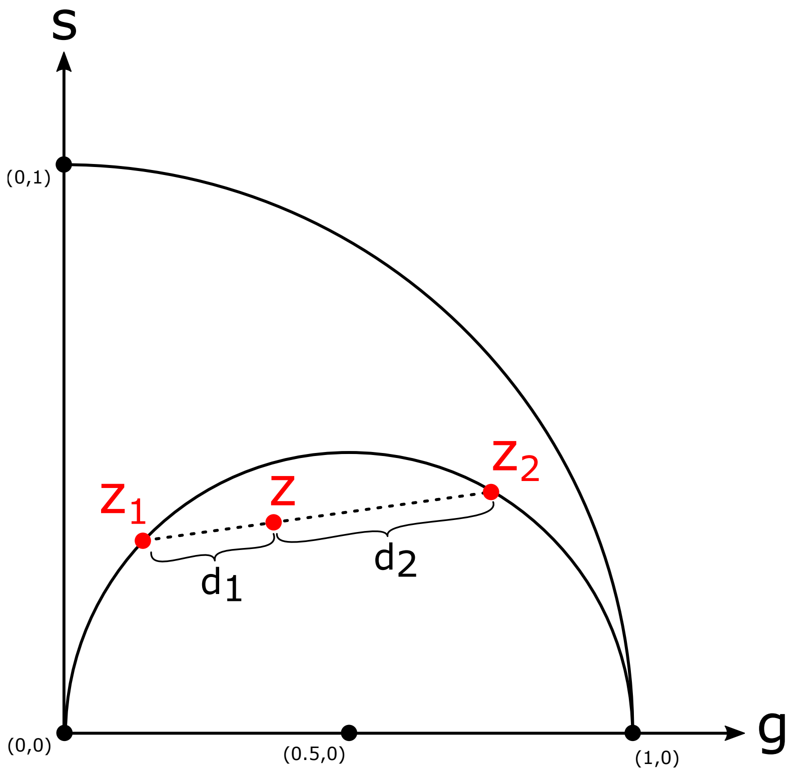


Figure 1.6: For a double-exponential decay, phasor z is located on the chord between the two single exponential components z_1 and z_2. The ratio of the two segments of the chord can be used to determine the ratio of two components.

response function (IRF) of the setup can be compensated for. As the phasor-method is essentialy a normalized discrete Fourier transform, deconvolving the IRF consists of a division with complex phasor z_{IRF}. This phasor can be obtained by measuring a single-exponential sample with known lifetime τ_{ref} and dividing the measured phasor z_{meas} with the theoretical z_{ref} mathematically obtained using τ_{ref}. Compensation phasors z_{IRF} need to be computed for every phasor frequency f and pixel of the detector.

Phasor-based FLIM can be further sped up by using fewer time bins and undersampling the decay. In this case, the single-exponential phasors are no longer on the semicircle centered at (0.5, 0) and appropriate corrections need to be applied. The effect of undersampling has been analyzed in [100] and [99].

Because of all of the above mentioned advantages, the phasor-based FLIM method was chosen as a target application for two of the designs presented in this thesis.

1.3 SPAD

1.3.1 Technology and implementation

A single-photon avalanche diode (SPAD) or Geiger-mode avalanche photo diode (G-APD) is a solid-state device consisting of a pn junction reverse-biased above its breakdown voltage. Under these conditions, a photon that is absorbed in the depletion region where the electric field is very high, will excite additional electrons due to its kinetic energy, which in turn will cause a self-sustaining avalanche through impact ionisations. The avalanche current will quickly rise to levels that can be damaging to the device, therefore quenching circuitry is normally connected in series with the diode. The simplest version consists of a ballast resistor, but if the durations of the quenching and recharging of the SPAD need to be precisely controlled, active implementations are employed. Figure 1.7 shows the SPAD operating principle.

In contrast to avalanche photo diodes (APDs) operating in linear mode, the magnitude of the avalanche current of a SPAD cannot be used to determine the photon count and is instead converted into a digital voltage, with virtually no readout noise. The leading edge of this pulse can be used to indicate the detection of even a single photon with sub-ns resolution.

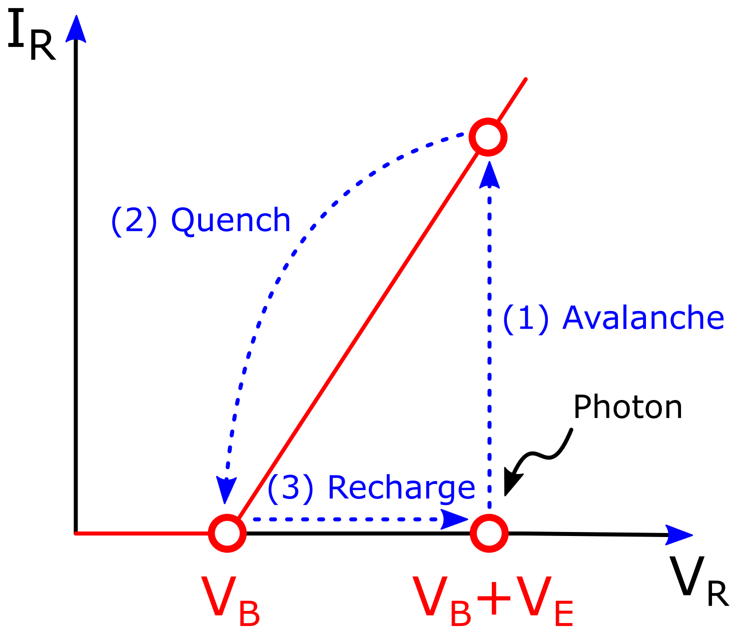


Figure 1.7: SPAD operation; V_B - breakdown voltage, V_E - excess bias voltage [101].

1.3.2 Metrology

The overall noise characteristic of a SPAD imager is dominated by dark counts due to the absence of the readout noise component. A dark count is a spurious avalanche pulse that was triggered in the absence of an incident photon through thermal generation, tunneling and/or trap-assisted processes. These effects are aggravated by increases in temperature or SPAD bias voltage. The average count rate under dark conditions normalized to one second is referred to as *dark count rate (DCR)*. If a SPAD pixel has a DCR at least two orders of magnitude greater than the median value of all the others, it is labelled as *hot* and usually excluded from further analyses [102], [103].

Both photon and noise generated events can in turn trigger a secondary avalanche, either in the same pixel or in the neighboring ones in the case of an array. The former is called afterpulsing and is characterized by the *afterpulsing probability*, while the latter constitutes crosstalk, it too reported as a *crosstalk probability*. The two metrics can be determined using inter-arrival statistics between photons in the same pixel for afterpulsing and in neighboring pixels for crosstalk [104]. An additional method for measuring crosstalk probability is by comparing the DCR of a neigbour of a hot pixel to the average DCR of the entire array [105]. In the case of gated SPAD sensors, the afterpulsing probability is significantly reduced by the long periods when the gate is turned off and the SPAD is insensitive. Conversely, there is no such effect on crosstalk which has to be reduced either through pixel design or by turning off possible aggressors, such as hot SPADs.

The light sensitivity of the SPAD is characterised by the *photon detection probability* (PDP), a quantity that represents the probability that an impinging photon will trigger an avalanche [106]–[108]. The metric is dependent on wavelength and can be defined as:

$$PDP(\lambda) = QE(\lambda) \times P_t, \tag{1.4}$$

where QE is the quantum efficiency of the detector and P_t is the avalanche triggering probability. By including the geometric characteristics of the SPAD, we can define the *photon detection efficiency* (PDE) as:

$$PDE(\lambda) = PDP(\lambda) \times FF, \tag{1.5}$$

where FF is the detector fill factor. PDP can be measured as the ratio between the number of detected photons and the number of incident photons on the entire pixel area. The FF is a purely geometric parameter that depends on the pixel pitch and the distance between two neighboring SPAD active regions. As SPADs operate at high reverse biases, premature edge breakdown needs to be prevented at the border of the main junction by using guard rings. In addition, strong contacts to the deeper side of the junction also need to be implemented, as shown in the example from Figure 1.8. These regions are insensitive to light and limit the maximum achievable fill factor of a round SPAD to 78.5%, however, the pixel pitch would need to be very large to achieve this [109]. A more convenient method is the use of on-chip microlenses [110]–[112], a method also used for one of the designs in this thesis.

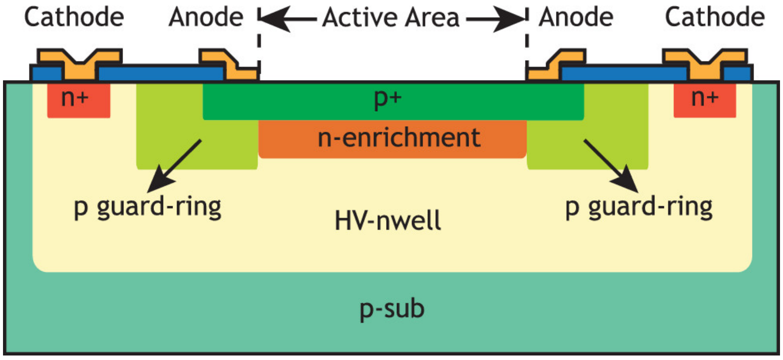


Figure 1.8: Example SPAD cross section: the active region is the junction between the p+ shallow diffusion and the n well, surrounded by the p-doped guard ring [113].

1.4 Research motivation

With current developments in large format SPAD imagers and the novel applications that make use of the single photon sensitivity and sub nanosecond resolutions of these sensors, the need to more efficiently manipulate the pixel data arises. Focus can be directed towards two major objectives: **frame rate increase** and **output data rate reduction**. The former will allow the monitoring or compensating for short time scale phenomena [114], [115] while the latter will reduce hardware and system constraints, as current SPAD imagers have already reached the maximum data rates of modern interfaces. Both problems can be solved through the use of on-chip processing elements and the adoption of computational imaging architectures. The digital nature of SPADs is well suited for these types of architectures as it does not require any analog-to-digital conversions and the pixels can be directly interfaced with the processing electronics.

The current trend with SPAD imagers is, as it was with the CIS counterparts, to reach larger, multi-megapixel formats [68] which implies that the computational architectures will have to be scalable, to not limit future developments, but also flexible to implement state-of-the-art computational imaging techniques. New, 3D-stacked

CMOS technologies allow the use of different nodes for the top, photodetector tier, and the bottom processing tier, so complex and scalable architectures can be designed without compromising the optical sensitivity. However, mature 2D technologies are still relevant and require different design paradigms, so research has to be conducted in parallel. In the end, a scalable and flexible architecture is needed to take advantage of SPAD-based imager characteristics, regardless of the technology used for fabrication.

1.5 Scientific contributions

The aim of this thesis was to develop sensor architectures for computational imaging that overcome current limitations of SPAD imagers and are capable of running at high frame rates while maintaining manageable output data rates. Three sensors have been designed and implemented in this thesis, all containing on-chip processing capabilities of varying complexities, from application-specific vector processing of pixel columns to fully customizable pixel clustering. The scientific contributions found in these architectures can be summarized as follows:

A token-based pixel readout was developed for SPAD imagers fabricated in 2D technologies and implemented in *kilo*Phase. As opposed to a conventional array readout, the token-based implementation only reads the pixels where a photon has been detected during the exposure, using dedicated channels for each row. As a result, readout time is proportional to the light levels of the scene, which depending on the application, can be sufficiently low to result in significant speed-up.

A scalable FLIM processing architecture that takes advantage of advancements in 3D stacking technologies which resulted in *Mega*Phase, the first 3D megapixel SPAD imager with on-chip computational imaging capabilities. The detector consists of multiple processing units operating independently, in parallel, each connected to clusters of SPADs that can be treated as individual binary pixels or binned together into pixels with larger dynamic range. The high degree of parallelism that this architecture employs results in an increase in frame rate of up to two orders of magnitude. Additionally, processing the SPAD data on chip on such a large scale reduces the output data rate by up to three orders of magnitude.

A fully reconfigurable processing architecture was developed and implemented in *Ultra*Phase as a proof of concept non application-specific computational imager. The design can be easily scaled to larger formats and can implement newly developed image processing algorithms without the need of hardware redesign. The result is an array of independent cores, each with connections to pixel front ends, each capable of running software programs with inter-core communication at over one hundred million operations per second.

1.6 Thesis structure

Chapter 2 presents *kilo*Phase, a 2D chip architecture developed for FLIM. The system architecture, trade-offs and characterisation results are all discussed. Chapter 3 is dedicated to *Mega*Phase, a megapixel imager in 3D CMOS also developed for FLIM, with the focus on system architecture and comparison with other large format SPAD imagers. Chapter 4 describes *Ultra*Phase, a fully reconfigurable imager with multicore processing and presents a couple of applications that can be easily implemented using the flexibility of the system. Finally, Chapter 5 summarizes the conclusions of the thesis and also contains future work proposals.

2 *kilo*Phase

*kilo*Phase is a 32 × 32 SPAD imager developed in 180 nm 2D CMOS technology, with a pixel pitch of 28.5 µm and a fill factor of 28%. The chip contains two accumulator banks each capable of storing one 10 bit image, 32 computational units that each implement one accumulate and two multiply-accumulate (MAC) operations, 3.072 kbit of ROM and 76.8 kbit of RAM. Readout of the SPAD array can be performed by using conventional column addressing or a novel token-passing technique that increases the frame rate by ignoring dark pixels, i.e., pixels that did not fire because no photons were detected or that are dead.

*kilo*Phase can execute the operations shown in equation (2.1) for each pixel, without the need of being read out and can simultaneously process one frame while exposing another.

$$\begin{cases} D = \sum_{m=1}^{128} \left(\sum_{n=1}^{1024} b[n] \right) \times \alpha[m] \\ I = \sum_{m=1}^{128} \left(\sum_{n=1}^{1024} b[n] \right), \end{cases} \tag{2.1}$$

where D is a complex output value, I is a real output equivalent to the intensity value of the pixel, b is the binary value of the pixel after exposure and α is a complex coefficient.

At the moment of writing this thesis, *kilo*Phase has been fabricated and tested. The functionality of the token-based pixel readout was successfully validated, together with all the on-chip processing circuitry. The pixel was completely characterized and the chip was operated as a 10-bit imager. The firmware required for FLIM operation is still in development.

2.1 Motivation

*kilo*Phase was designed as a proof of concept for real-time computational imaging architectures when only 2D integration technologies are available. The improvements brought forward by this implementation target a frame rate increase through the reduction of the SPAD array readout and off-chip communication along with on-chip data processing.

The objective was achieved by following a set of design guidelines:

- Only pixels with valid information are read.
- All binary frames are accumulated into grayscale frames on chip.
- Grayscale frames are further processed on chip.
- Coefficients needed for processing are loaded at startup.
- Processing of a frame and accumulation of another are done simultaneously.

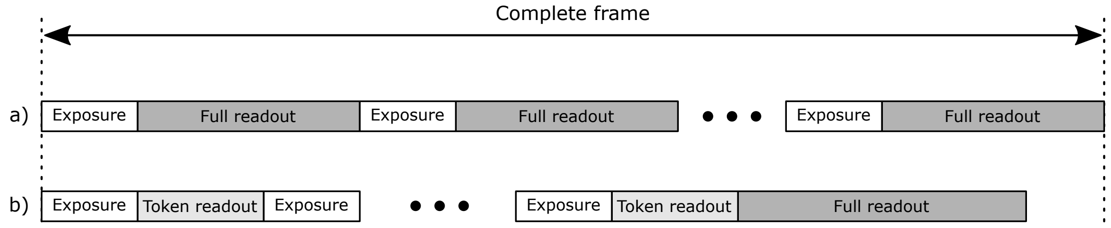


Figure 2.1: Comparison between the readout schemes of a conventional SPAD imager operating with global shutter a) and *kilo*Phase b). In the case of the latter, each binary exposure is followed by a token-based readout that on average is faster than the conventional one. A full readout is only required at the end of the acquisition.

The resulting chip operation no longer requires the off-chip readout of binary frames and instead condenses all the information into a single processed frame, read out at the end of the computation. Figure 2.1 shows a comparison between conventional gated SPAD imagers like the ones presented in [66], [71], [116] operating with a global shutter and *kilo*Phase. Assuming that the conventional chips are used for gated FLIM (as described in Chapter 1.2.1), we can compute the total time T_{frame} required to acquire one FLIM frame as:

$$T_{frame} = (N_{EXP} \times T_{laser} + T_{binaryreadout}) \times N_{ACC} \times N_G, \tag{2.2}$$

where N_{EXP} is the number of openings of the gate signal during the exposure of one binary frame, N_{ACC} is the number of binary frames being accumulated into a grayscale image, N_G is the number of gates used for the FLIM computation, T_{laser} is the pulse repetition period of the laser and $T_{binaryreadout}$ is the total time needed for the readout of a binary frame.

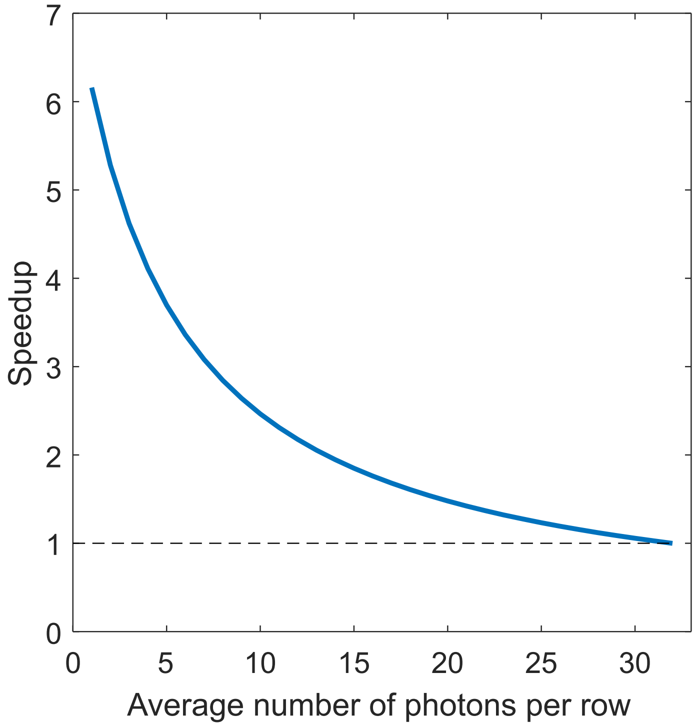


Figure 2.2: Theoretical maximum speedup of *kilo*Phase compared to conventional SPAD imagers when operating in gated-FLIM with 128 gates

For *kilo*Phase, two readout durations need to be taken into account: the time required to transfer the binary frame into the on-chip accumulator bank $T_{binaryreadout}$ and the chip readout $T_{readout}$. The equation for T_{frame} becomes:

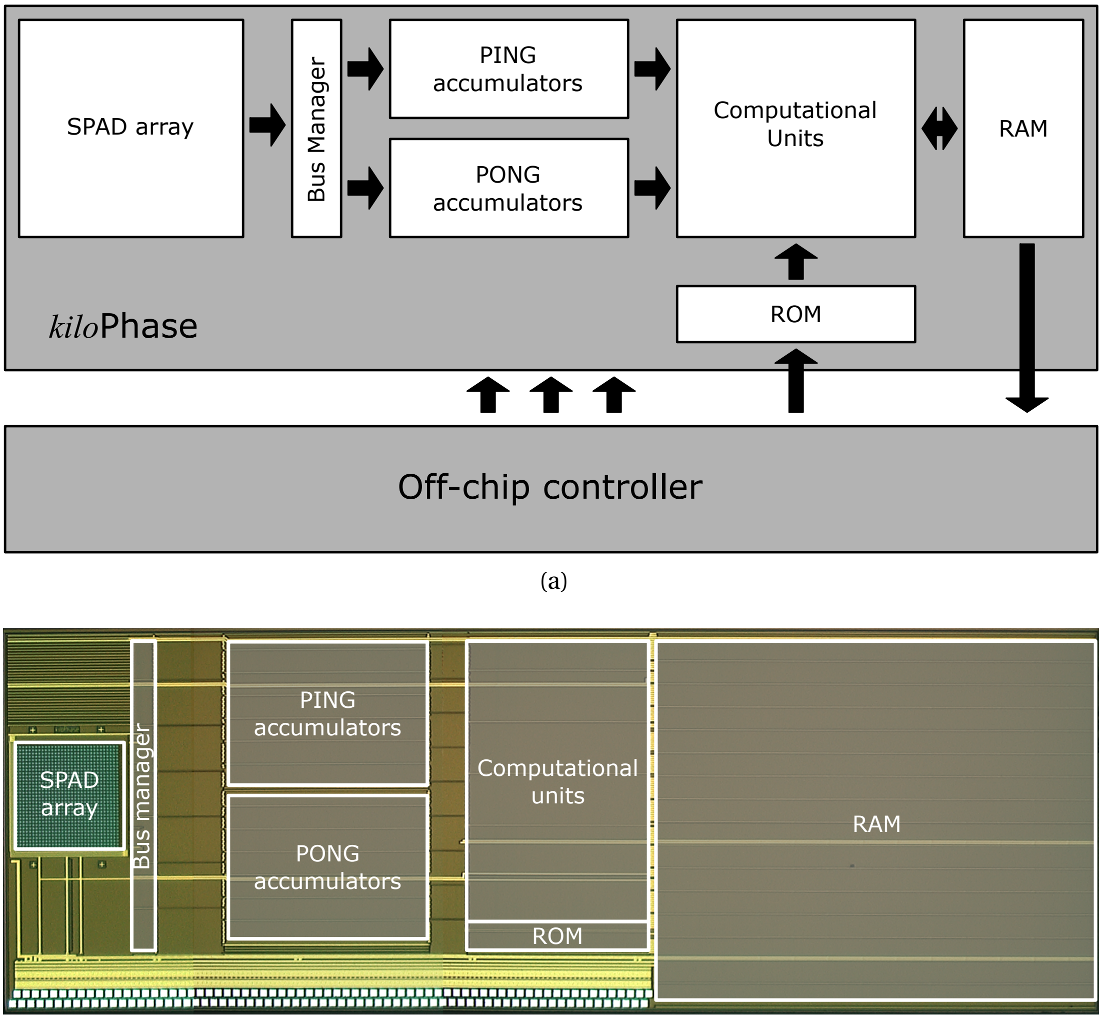
$$T_{frame} = (N_{EXP} \times T_{laser} + T_{binaryreadout}) \times N_{ACC} \times N_G + T_{readout}. \tag{2.3}$$

In practice, $T_{readout} >> T_{binaryreadout}$ and the latter can be computed using:

$$T_{binaryreadout} \leq N_{col} \times T_{CLK}, \tag{2.4}$$

where T_{CLK} is the period of the system clock and N_{col} is the number of columns in the detector. In the case of conventional detectors with column-parallel readout, the equation is an equality. However, in *kilo*Phase each row is read independently and only the pixels that have fired are used. Similar architectures have been presented in literature, however, this is the first autonomous implementation and the first one to have pixel level granularity [117], [118]. As a result, depending on the illumination, a binary frame can be read out in fewer clock cycles. This leads to the maximum increase in gated-FLIM frame rate as shown in Figure 2.2 given for a 128 gate configuration.

2.2 Processing flow



(a)

(b)

Figure 2.3: (a) *kilo*Phase block diagram consisting of the SPAD array, a Bus Manager for token-passing readout, two accumulator banks, a group of 32 Computational Units and memory blocks. The chip is controlled by an external circuit implemented on FPGA. (b) The same elements highlighted over the chip micrograph. Die size is 9.5×3.3 mm^2.

A high level block diagram of *kilo*Phase is presented in Figure 2.3. The seven functional blocks and the communication between them are controlled by a separate external circuit implemented on FPGA. This configuration allows for a high level of customisation but comes at the cost of simplicity of the chip interface which requires 29 control signals.

Normal sensor operation consists of three stages: exposure, computation and readout. The first two can function simultaneously through the use of the PING-PONG accumulator architecture, but chip readout must be performed separately.

Exposure begins with N_{EXP} openings of the gate signal that result in a 32 × 32 binary image being stored in the *SPAD array* in-pixel memory. This data is then transferred to one of the accumulators via the *Bus manager* either one column at a time or through token-passing (subsection 2.3.2). The cycle is repeated N_{ACC} times until a grayscale image of the desired bit depth is formed in the accumulator. Total exposure T_{EXP} for one gate configuration can be computed using equation (2.5) as:

$$T_{EXP} = N_{ACC} \times N_{EXP} \times W_G, \tag{2.5}$$

where W_G is the width of the gate signal. A new exposure cycle can begin by switching to the second accumulator bank, which leaves the data that has already been collected intact.

The computation stage lasts exactly 32 clock cycles T_{CLK} during which the 32 *computational units* will process the data stored in the accumulator bank, one array column at a time, together with the corresponding values from the RAM and ROM. Two MAC and one accumulation operation are simultaneously computed for each pixel p at every iteration n of the computation stage:

$$\begin{cases} D_G^p[n] = D_{ACC}^p \times \alpha_G[m] + D_G^p[n-1] \\ D_S^p[n] = D_{ACC}^p \times \alpha_S[m] + D_S^p[n-1] \\ D_I^p[n] = D_{ACC}^p + D_I^p[n-1], \end{cases} \tag{2.6}$$

where D_G and D_S are the 29 bit fixed point real and imaginary parts of the complex pixel value, D_I is the 17 bit integer intensity component, D_{ACC} is the 10 bit integer value stored in the accumulator and α_G and α_S are 12 bit fixed point real and imaginary parts of the complex coefficient stored in the ROM at address m. All fixed point values are signed and have 10 fractional bits. D_G, D_S and D_I are stored in the RAM and reset to 0 at startup and after chip readout, while D_{ACC} is reset at the begining of the exposure cycle. α_G and α_S are loaded at startup. The widths of the variables were chosen to accommodate as many accumulations as possible while respecting the area constrains of the design. The fractional representation results in a relative rounding error of less than 2% when used to store sine and cosine values.

As long as the duration of the exposure is longer than $32 \times T_{CLK}$, there will be no dead time of the detector, as the two accumulator banks allow simultaneous processing of the previous frame while exposing the current one. The exposure and computation stages can be repeated multiple times until the desired processing is completed, the only limitations being the ROM size of 128 coefficients and the bit widths of the accumulator and RAM. For example, in the case of FLIM, *kilo*Phase can accommodate 128 gate positions with 10 bit grayscale images for each, or 32 gate positions at 12 bit.

Once all the necessary exposure and computation cycles are performed, the RAM can be read out in $1024 \times T_{CLK}$ using a 75 bit parallel bus. The readout stage is incompatible with the normal operation of the chip due to pad multiplexing, therefore, the chip must be kept idle.

2.3 Architecture

2.3.1 Pixel

The pixel schematic implemented in *kilo*Phase is shown in Figure 2.4 with the associated layout in Figure 2.5. A total of 10 NMOS transistors are used, with both thin and thick oxide variants. The SPAD structure and 28.5 µm pixel pitch is identical to the one presented in [37] in order to allow the reuse of an existing microlense design for increasing the 28 % native fill factor.

Cascode transistor T_1 is used to extend the bias voltage range of the pixel and increase the PDP as demonstrated in [119]. Passive quenching of the SPAD is implemented with transistor T_2 and tunable analog voltage V_Q, while T_3 is used for the clock-driven active recharge controlled by digital signal RCH. When an avalanche is formed in the SPAD, the voltage at node A will rise and drive transistor T_4 which in turn will bring node B to ground. Practically, T_4 serves as a voltage level shifter between the thick oxide transistor section (T_1 to T_4) that operates at 3.3 V and the rest of the 1.8 V circuitry. When the $GATE$ signal is asserted, transistor T_5 shorts together nodes B and C which will result in two possible outcomes: if T_4 is also active, node C will be pulled to ground and cause the memory cell formed by T_6, T_7, T_8 and T_9 to toggle. Otherwise,

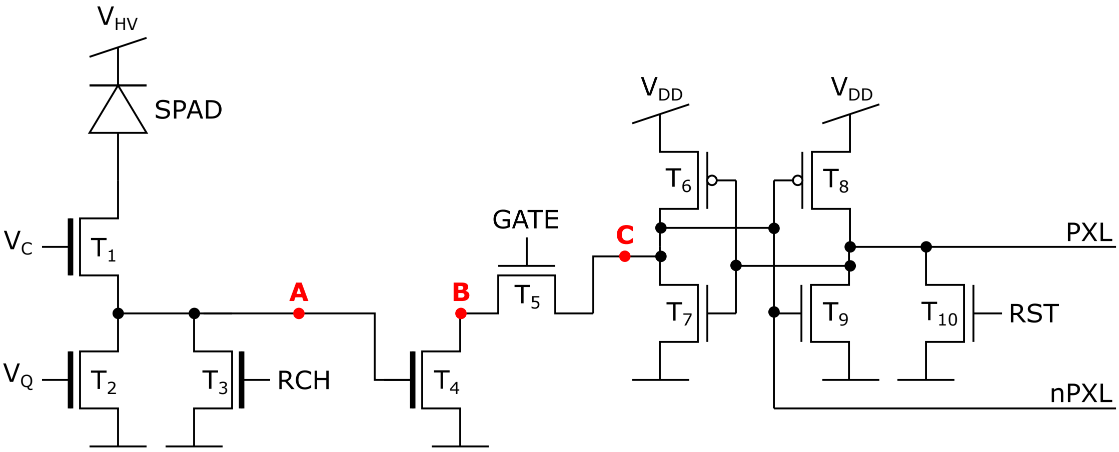


Figure 2.4: *kilo*Phase pixel schematic containing 4 thick oxide and 6 thin oxide NMOS transistors. Cascode transistor T_1 is used to extend the SPAD bias voltage range. T_2 acts as a passive quenching method and T_3 implements a clock-driven recharge scheme. A pair of back-to-back inverters that can be reset by T_{10} form a static memory cell. Transistor T_5 is used to gate the pixel.

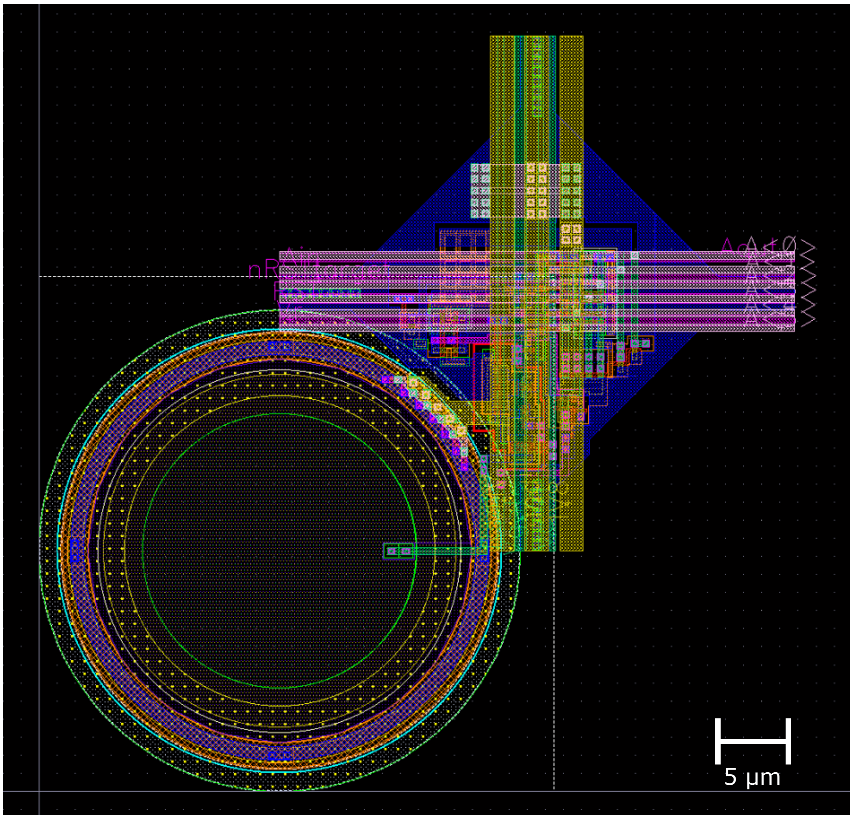


Figure 2.5: *kilo*Phase pixel layout. The NMOS transistors are all located in the upper right corner of the SPAD. Metallisation is patterned so that the pixel can be abutted to form the desired array with 28.5 µm pitch.

node C will not change because node B will be at high impedance and the memory cell will remain in the reset position. At the beginning of each exposure cycle, RCH is pulsed right before the arrival of the first $GATE$ which assures that any avalanches that have occurred before the arrival of the gate signal will be ignored. Transistor T_{10} is used to reset the memory. Both the pixel output signal PXL and its inverted value $nPXL$ connect to the token-passing cell presented in the following subsection.

2.3.2 Token-passing readout

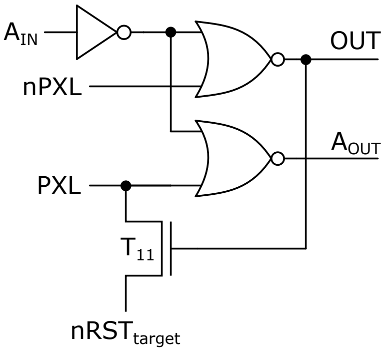


Figure 2.6: *kilo*Phase token-passing schematic consisting of two NOR gates, an inverter and an NMOS transistor. The pixel output PXL is not allowed to pass unless token signal A_{IN} is high. In this state, if PXL is driven high, A_{OUT} will be driven low and cause all further token-passing cells to restrict bus access for their corresponding pixels. T_{11} can be used to perform a targeted reset of the pixel if and only if the pixel currently has access to the output bus.

The token-passing readout mechanism used to reduce the array readout time by skipping pixels that have not fired is implemented independently for each row of the array. Each pixel is assigned a priority, starting with the leftmost pixel in the row that has the highest and decreasing all the way to the lowest priority rightmost pixel. Only the highest priority pixel that has detected a photon can access the row readout bus. In addition, this is also the only pixel that will react to a targeted reset signal $nRST_{target}$ and will clear its memory and pass the bus access *token* to the next lower priority pixel that requests it.

Figure 2.6 shows the schematic for the token-passing logic circuitry. The pixel output signal PXL and its inverted value $nPXL$ connect to two NOR gates along with the negated value of token signal A_{in}. When A_{in} is low, both OUT and A_{OUT} will be kept low, regardless of the value of PXL. Therefore, the pixel loses access to the bus by having its output signal gated and in turn also blocks the next pixel by not having any token to pass. When A_{in} is high, the top NOR gate acts as an inverter and the

OUT signal has the same value as PXL; in other words, the pixel is granted access to the bus because it holds the token. If the access is needed, i.e. PXL is high, then the bottom NOR gate will drive A_{out} low which will block the next pixel's access to the bus. If, however, there has been no photon detection and PXL is low, the token is passed to the next pixel by driving A_{OUT} high. If the pixel has the token and needs access to the bus, gating transistor T_{11} will be on, driven by the output signal, and $nRST_{target}$ can be used to reset the in-pixel memory by forcing PXL low.

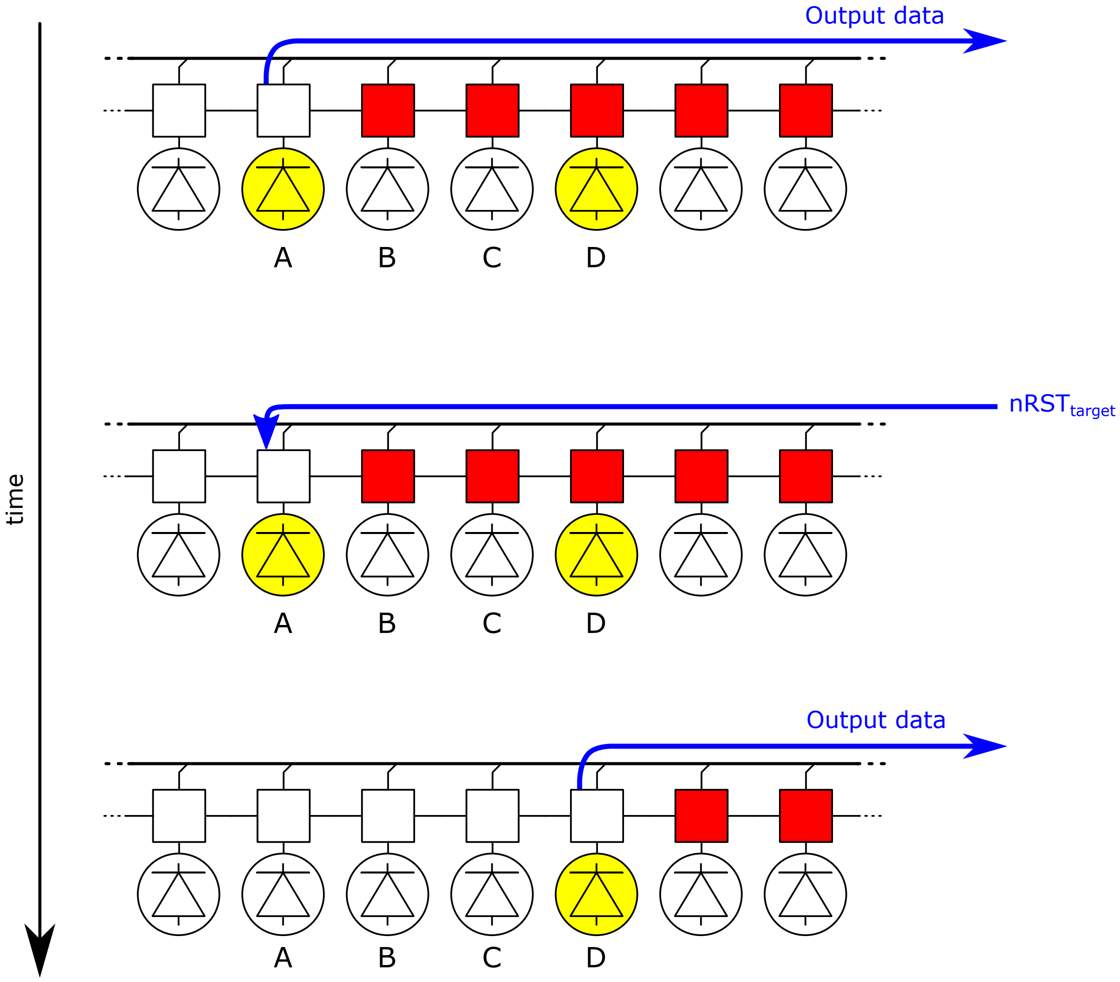


Figure 2.7: *kilo*Phase token-passing principle. Three states of the same pixel row at different moments in time are shown, chronologically from top to bottom. Both pixels A and D have detected an avalanche, but because the former has priority it receives access to the data bus and blocks the remaining pixels. After the targeted reset signal is received, pixel A will reset and release the bus. The token is passed directly to pixel D as it is the next highest priority pixel with a detected avalanche.

Figure 2.7 is a representation of the token-passing principle applied for the readout of one row in the case of two detected photons. Both pixel A and pixel D have detected a photon. However, pixel A has the highest priority and therefore has access to the output bus and holds the token. When the $nRST_{target}$ signal is asserted, only pixel A will respond by resetting its internal memory and passing the token to pixel B. Because pixel B does not need bus access, it will automatically pass the token to pixel C, which for the same reasons, will pass it to pixel D. In this case, bus access is needed and the token is kept until the arrival of another targeted reset. The entire process takes place after the exposure, when the pixel gate is closed and the output cannot change. As a result, there is no risk of corrupting the readout.

2.3.3 Bus manager

The role of the Bus manager circuit block is to convert the pixel output information that comes in the form of individual addresses for every row into the 1024 bit input to the accumulator banks. It also needs to accommodate both token-passing and column addressing readouts.

Each pixel row is connected to the circuitry shown in Figure 2.8. In token-passing readout, when a pixel has access to the bus, it will transmit its 5 bit address A to the bus manager where it will be decoded into a one-hot 32 bit representation using a 6-to-32 decoder. The redundant use of 6 bits comes from the fact that all pixel addresses are offset by 1 in order to use bus address 0 as a representation of the free state of the bus. When the rising edge of CLK_{fast} arrives, the output oH of the bus manager is updated with the one-hot representation of the current token holder pixel through the use of the OR gate. The process repeats when the token is passed to another pixel and ends when the bus is free, at which point oH stores a complete representation of all the pixel states in the row. Asserting CLK_{frame} clears oH and prepares the circuit for a new readout cycle.

When $MODE$ is at logic zero, the bus manager operates in column addressing readout. Under these conditions, the output of the in-pixel memory is connected to $A[4]$ by a column select signal. The values for each pixel are then serially loaded into the bus manager at every rising edge of CLK_{fast}. Readout is complete after 32 CLK_{fast} cycles.

2.3.4 Accumulator bank

Each accumulator bank consists of 1024 10 bit counters arranged in a 32 × 32 grid that mirrors the pixel array. Each row of 32 counters shares a 10 bit output bus, access to which is controlled using a 5 bit selector signal common to all the rows. The output buses for each row are concatenated into a 32 × 10 bit bank bus.

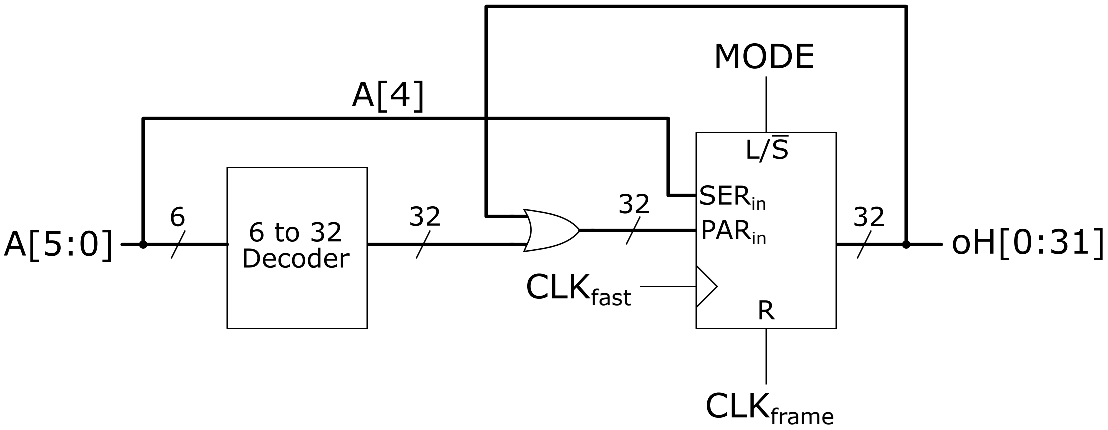


Figure 2.8: *kilo*Phase Bus manager circuit schematic for one row of the array. The 6 bit pixel output bus A containing the column address of the current token holder is decoded into a 32 bit one-hot representation and then combined with the Bus Manager output with an OR gate. CLK_{fast} is used to update the output with the new value while CLK_{frame} acts as a reset signal. If $MODE$ is held low, conventional readout is used and the oH output is updated serially by shifting the values on bit 4 of the input bus.

Each individual counter can be enabled using the corresponding bit from a 1024 bit input bus, considering that the top left counter has index 0 and the bottom right one has index 1023. A global synchronous reset signal is used to clear all the counters in the accumulator banks.

The 1024 bit inputs to the two accumulator banks are driven by a 1-to-2 demultiplexer (DMUX) which in turn is driven by the Bus manager. The two 32 × 10 bit bank outputs are merged using a 2-to-1 multiplexer (MUX) and then serve as inputs to the computational units. The DMUX and MUX control signals are independent from each other, therefore, the accumulator banks can operate in three modes: PING-PONG, PING only and PONG only.

2.3.5 Computational unit

A total of 32 computational units are used in *kilo*Phase, one for each row in the array/accumulator. Using the accumulator selector signal, the pixel data corresponding to an entire array column is presented to the computational unit inputs and processed in parallel.

Figure 2.9 shows the schematic for a single computational unit composed of two

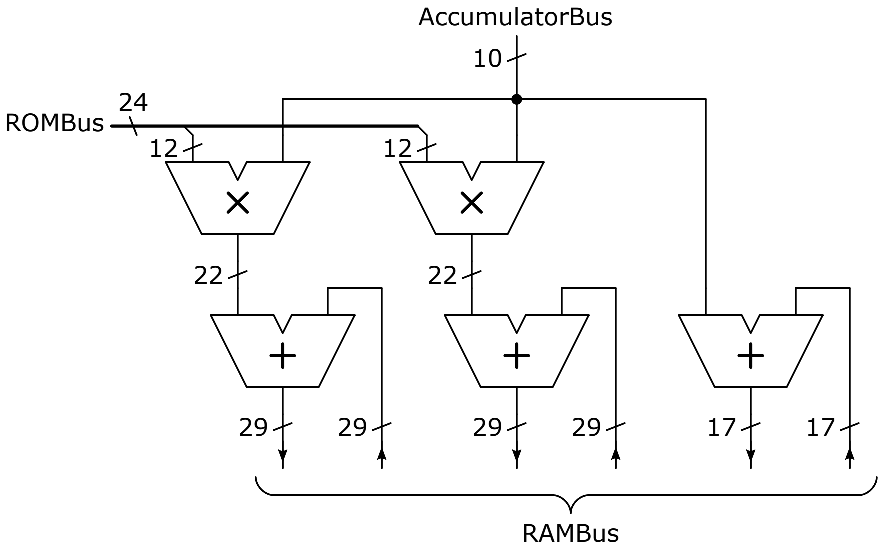


Figure 2.9: *kilo*Phase computational unit schematic showing the two 22 bit signed multipliers and the 29 bit, 29 bit and 17 bit accumulators. The operands for the MAC and accumulate operations originate from the upstream pixel accumulators and downstream RAM. The constants required for the two MAC operations are read from a common ROM.

multipliers and three adders. The operations are signed and represented using a fractional fixed point format with 10 fractional bits. The only exception is the standalone 17 bit integer adder used for intensity imaging.

The 10 bit pixel data stored in the accumulators is provided via the $AccumulatorBus$. The value is multiplied with two 12 bit constants stored in ROM and then accumulated with a 29 bit value read from the RAM. A third adder circuit performs a 17 bit accumulation operation with the pixel data. The results are written back to the RAM using the $RAMBus$.

The $ROMBus$ used to transmit the two multiplication constants is 24 bits wide and is split between the two multipliers in order to represent a complex number. The $RAMBus$ consists of a 75 bit input and a 75 bit output signal.

2.3.6 RAM bank

Figure 2.10 shows a high level schematic of the RAM bank implemented in *kilo*Phase. The total available memory is 1024 × 75 bit, organized as 32 *slices* of 32 *units*, each composed of 75 1-bit *cells*. A cell consists of two back-to-back inverters connected to a *slice bus* through two tri-state buffers. The bidirectional slice bus connects to a computational unit through a *slice interface* block that splits it into a latched *read bus* and a buffered *write bus*. The other end of the slice bus is buffered to a common *OUTPUT BUS* that is used for external readout of the RAM.

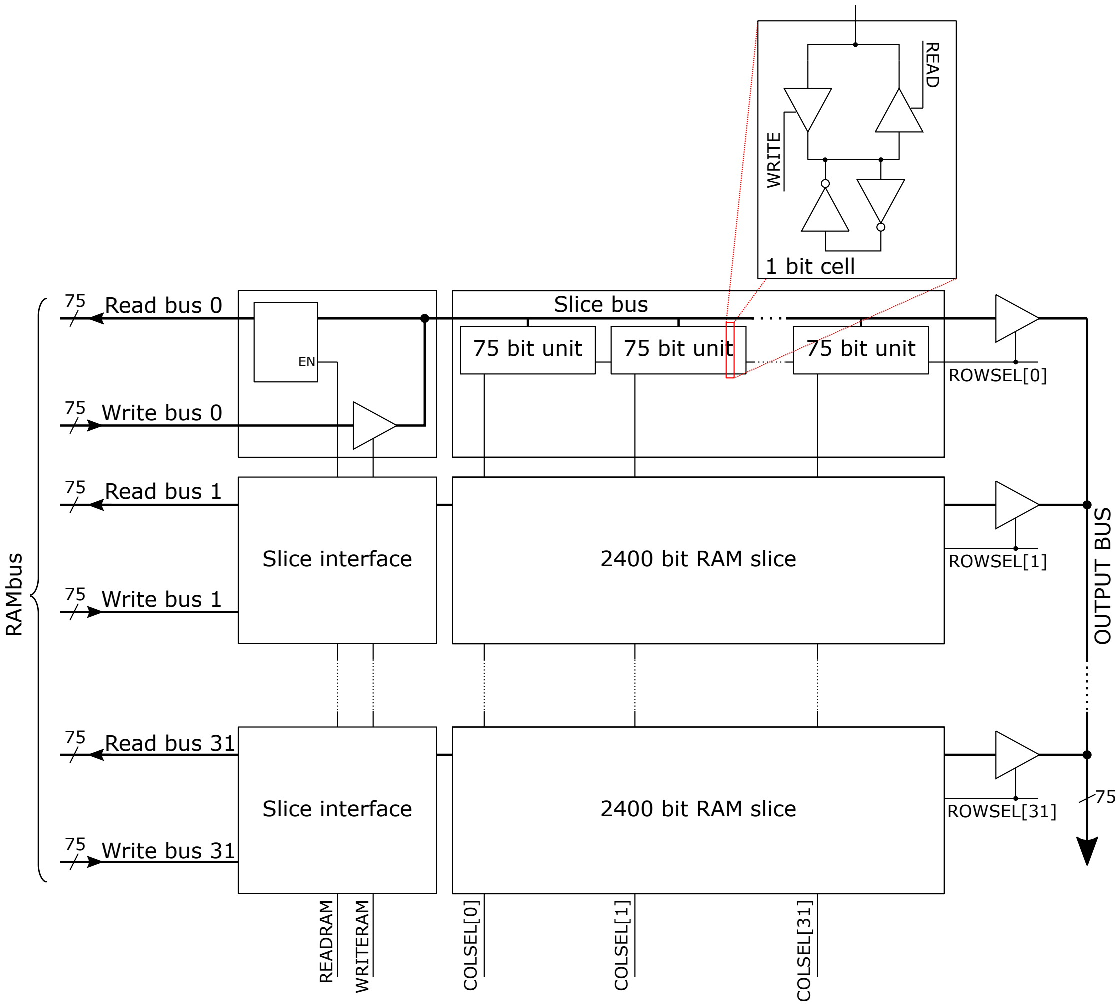


Figure 2.10: *kilo*Phase RAM schematic illustrating the complex structure consisting of 32 slices of 32 units. Reading and writing operations are performed using 32 bidirectional buses of 2 × 75 bit. During chip readout, the RAM is read using a row and column addressing scheme, one 75 bit unit at a time, on a separate 75 bit tristate bus.

During the computation stage, one unit from each RAM slice is granted access to the corresponding slice bus and its value is latched into the read bus. At the next clock cycle, the computational unit output which is present on the write bus is propagated to and then stored into the aforementioned slice. The two steps are repeated until all 32 units from the slice have been processed. This arrangement allows the simultaneous processing of an entire 32 pixel column of the array.

During the readout stage, the 1024 RAM units are sequentially given access to the OUTPUT BUS using a row and column addressing scheme. As this procedure also utilizes the slice bus, the readout is incompatible with the computational stage and the chip needs to be kept idle.

2.4 Implementation

*kilo*Phase was implemented as distinct sections that were then manually integrated to form the complete system. The design for the pixel array was done by hand while all of the remaining blocks shown in Figure 2.3 were synthesized with Cadence RTL Compiler and routed with Cadence Encounter.

Pixel

The various transistors from the pixel were sized using SPICE simulations were the SPAD was replaced with an equivalent model like the one shown in [120]. With the layout completed, SPICE simulations with the extracted parasitic components were run in multiple library corners to ensure the validity of the circuit.

When performing the parasitic extraction, the SPAD had to be replaced with a black box as the layout versus schematic (LVS) step of the process couldn't recognize the structure. This is the only point in the design were manual verification had to be used to confirm the final layout.

Gate and recharge signals

The gate and recharge signals propagate across the pixel array in two orthogonal directions in order to reduce the coupling between them and maintain good signal integrity as the shape of the gate is essential to the operation of the chip. This propagation scheme, shown in Figure 2.11, results in a diagonal skew in the measured gate signal as seen in Section 2.5.7, but the small size of the array reduces the significance of the effect.

The gate and recharge signals connect to the pixel array through two $1-to-32$ trees that minimize the skew between the propagation channels. The schematic is shown in Figure 2.12. The trees are built using custom inverter cells that are sized to maintain the propagation delay through the tree as short as possible when driving a 500 fF load on each output. The load was estimated based on the post layout parasitic report.

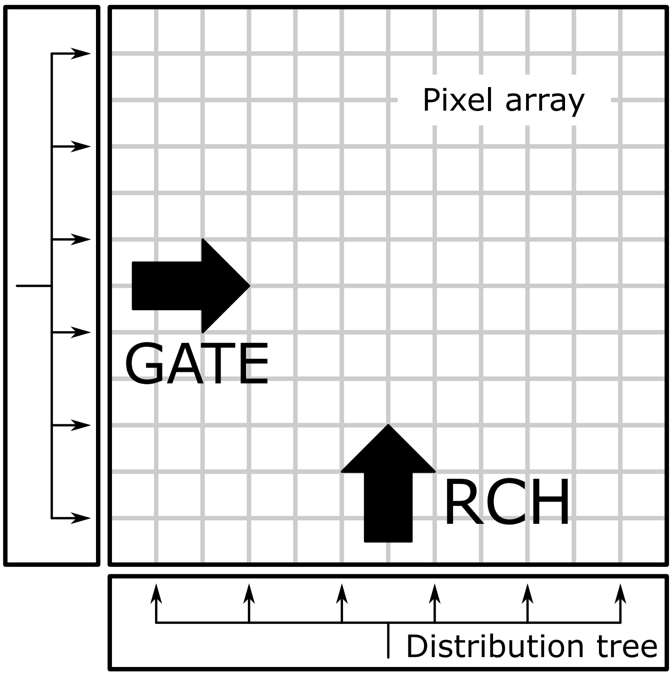


Figure 2.11: *kilo*Phase gate and recharge propagation scheme across the array. The quadrature arrangement reduces the coupling effect between the two signals.

The final tree design was validated through post layout SPICE simulations in a test bench that used a VerilogA component to act as a load and to measure the delay and skew across all 32 channels simultaneously. The simulations were run in 5 separate transistor library corners: Typical, Fast-Fast, Slow-Slow, Fast-Slow and Slow-Fast.

The power distribution network for the two trees was designed to have low impedance to reduce the voltage drop seen by the inverters in the middle of the tree. A total of 11 pF of decoupling capacitance was added to each tree, in between all the standard cells.

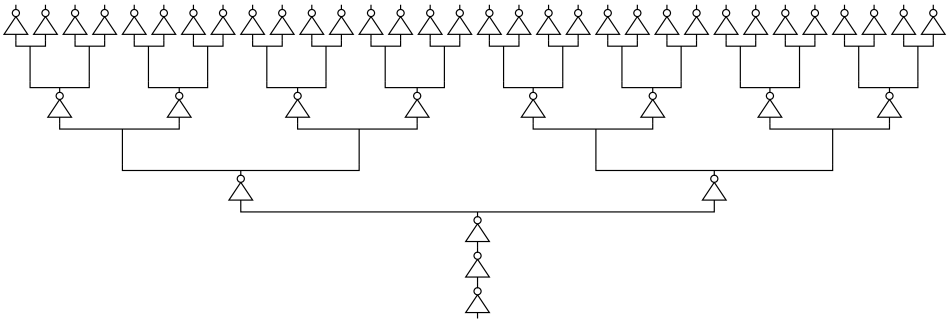


Figure 2.12: *kilo*Phase gate and recharge tree. The inverters were sized to drive a 500 fF load at each of the 32 outputs with minimal propagation delay.

Timing constraints

As each *kilo*Phase module was synthesized and routed separately, they each had their own timing constraints. The accumulator banks, ROM and RAM were designed for an operation at 200 MHz. The combinational components, namely the MUX, DMUX and computational units were constrained with a virtual clock of 300 MHz. After timing closure was confirmed, the histogram of signal slacks was examined to ensure that the designs were not over or under constrained.

It is important to mention that because the modules were synthesized separately, the input and output constraints had to be written in such a way as to account for the extra delay and parasitic loads that result from the manual integration. For this purpose, all of the designs assumed a minimum sized inverter as their input diver and the input of an 8× inverter as their load. The RAM was an exception as it drives a 75 bit output bus that is significantly longer than the other interconnects. In this case, the load was estimated using the interconnect parasitic capacitances reported by the foundry.

Power distribution

The power grid was routed on the top two available metals that are significantly thicker compared to the rest and exhibit lower resistance. Each *kilo*Phase block was surrounded with a wide power ring that was connected with every standard cell row to assure uniform power distribution. The rings were then stitched together at the top hierarchical level with horizontal and vertical metal stripes as shown in Figure 2.13. The stripes then directly connect to the wide pad ring metals.

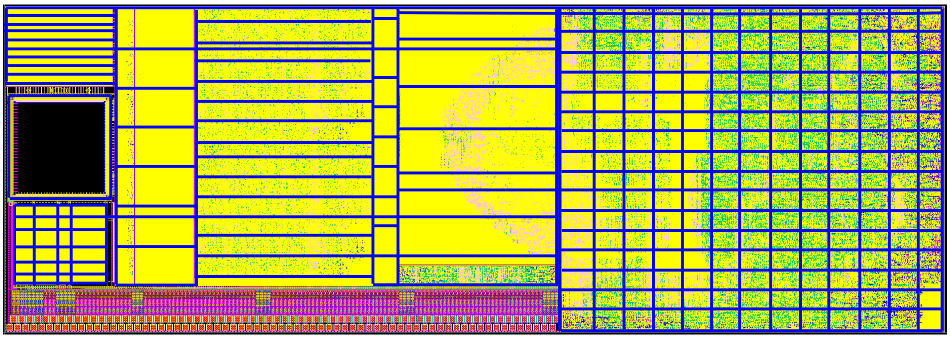


Figure 2.13: *kilo*Phase power distribution network (blue) and decoupling capacitors (yellow).

All of the empty spaces between the standard cells were filled with decoupling capacitors shown in Figure 2.13 highlighted in yellow. The core voltage rail decoupling capacitors total 30.5 nF.

The SPAD V_{HV} rail could not be decoupled using MOS capacitors because the required voltage exceeded the technology capabilities. A different approach was used where the track-to-track parasitic capacitance of the thick top metals was exploited by routing the supply in close proximity to a ground connection, thus creating decoupling capacitors.

Dummy generation

Dummy generation was performed on the entire design once the integration was completed. The areas above the SPADs contained blocking layers to prevent the dummy patterns from obstructing the light sensitive areas. Manual tweaking was required to ensure that the pixel area passed the minimum density requirements because of the geometric constraints that made it difficult for the tool to successfully generate the patterns.

A special situation was encountered when generating the top metal patterns. The structure of the power supply distribution network and the large spacing requirements of the top metal geometries resulted in the tool failing to make the design pass the density check. As a workaround, the rule file used for metal generation was manually changed to force the tool to use the top metal signal layer instead of the dummy one to create the patterns as this layer allows smaller structures that can fit in the required gaps.

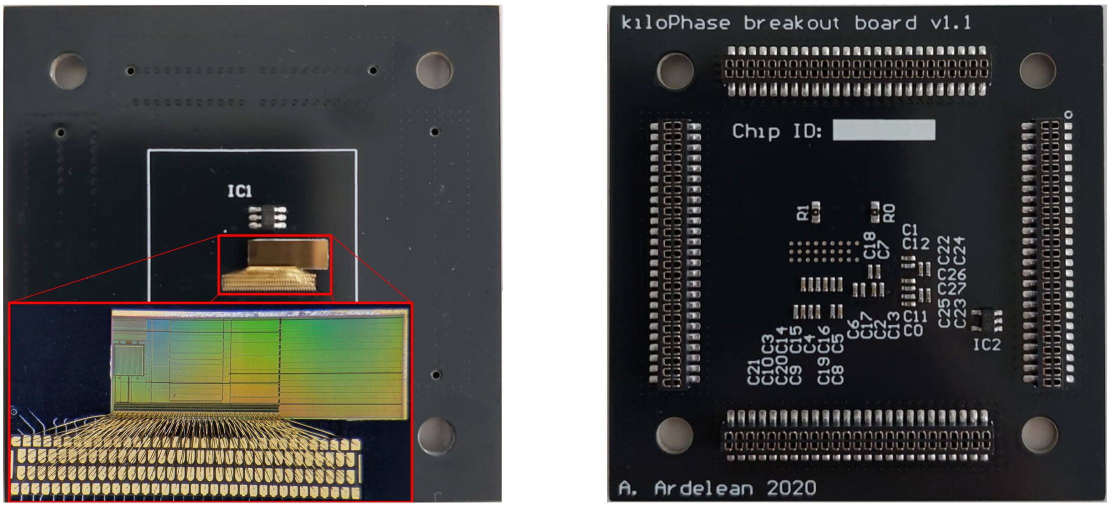
Validation

The design was validated by running digital simulations in ModelSim. The Verilog netlist of each module was exported from Cadence Encounter together with the .sdf file containing the net propagation delays for the Slow-Slow, Fast-Fast and Typical transistor corners. The pixel circuit was replaced with a custom Verilog model based on SPICE simulations.

2.5 Characterization

2.5.1 Camera system

*kilo*Phase was bonded in a chip on board (CoB) configuration because of the small 40.32 µm staggered pad pitch and to facilitate power dissipation by using the larger PCB copper planes. A 3D-printed lens mount was designed to fit onto the PCB and act as a rigid mounting point for the system. Figure 2.14 shows the CoB PCB without the lens mount. The four layers of angled bonding wires are clearly visible in the bottom left of the chip. The 32 × 32 SPAD array can be seen in the center of the PCB.

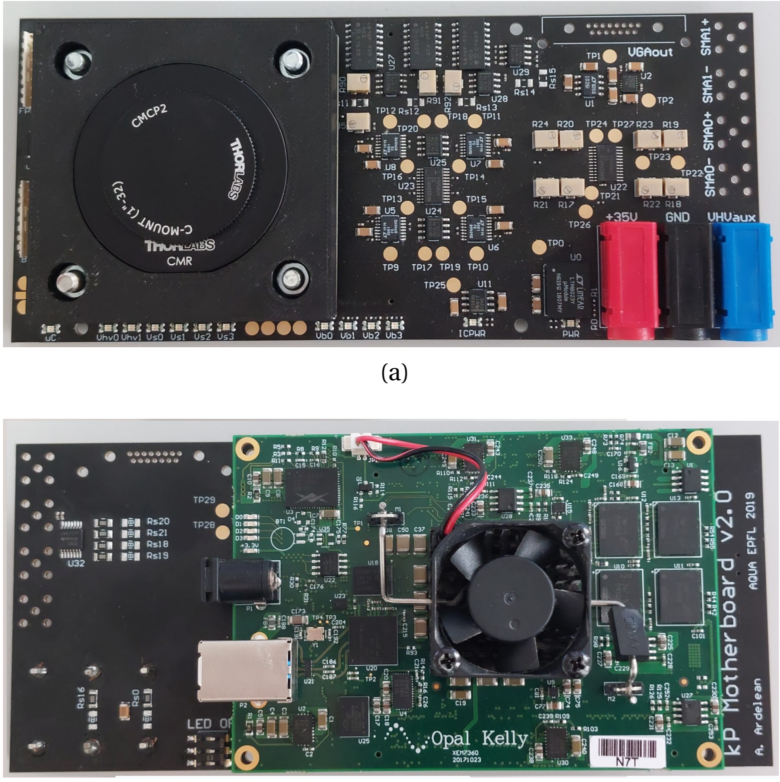


(a) (b)

Figure 2.14: *kilo*Phase CoB PCB (a) top and (b) bottom. The bonding wires are visible in the inset. The rectangular SPAD array can be seen on the left side of the chip, and is positioned to be in center of the PCB. The component next to the chip is a temperature sensor coupled with the main heat sink copper plane. All the decoupling capacitors are placed on the bottom side, as close as possible to the bonding wires. The board is 6 × 6 cm^2.

An Opal Kelly XEM7360 [121] module with a Xilinx Kintex-7 FPGA is used to generate the gate, recharge and control signals and to readout the data. A total of 16364 75 bit frames can be stored in the FPGA RAM before all the data is sent to the PC via a USB3.0 connection.

The CoB PCB interfaces with the Opal Kelly module through a custom motherboard PCB that also provides digitally adjustable power supply channels following a specific power up sequence and secondary functions such as die temperature and current consumption measurements. Figure 2.15 shows the complete assembly.



(a)

(b)

Figure 2.15: *kilo*Phase system (a) top and (b) bottom. The custom motherboard PCB connects *kilo*Phase to the FPGA board, provides the necessary power rails and monitors chip current consumption and temperature. A 3D-printed mount is used to connect the system to the optical setup. The entire assembly is 16 × 7 × 6 cm^3.

A *Dashboard* software written in Python 3.6 is used to program the FPGA, run the measurement according to user defined parameters and then read and process the output data. External scripts can be run to facilitate measurement repeatability and validity.

2.5.2 Breakdown voltage

Breakdown voltage measurements were performed by sweeping the SPAD bias voltage in 10 mV steps between 23 V and 25 V and acquiring dark frames at each value [122]. In total, 128 10 bit frames acquired with a gate width of 13 ns for each binary frame were accumulated for each bias voltage. The breakdown voltage of a specific pixel was determined as being 0.7 V less than the lowest bias voltage at which at least one count was detected. The subtracted voltage corresponds to the nominal threshold voltage of the thick oxide transistor used in the pixel.

Figure 2.16 shows the resulting breakdown voltage histogram for a single die. The average breakdown voltage was determined to be 22.58 V with a standard deviation of 39.7 mV. The results are in accordance with [37] which has the same SPAD and array size.

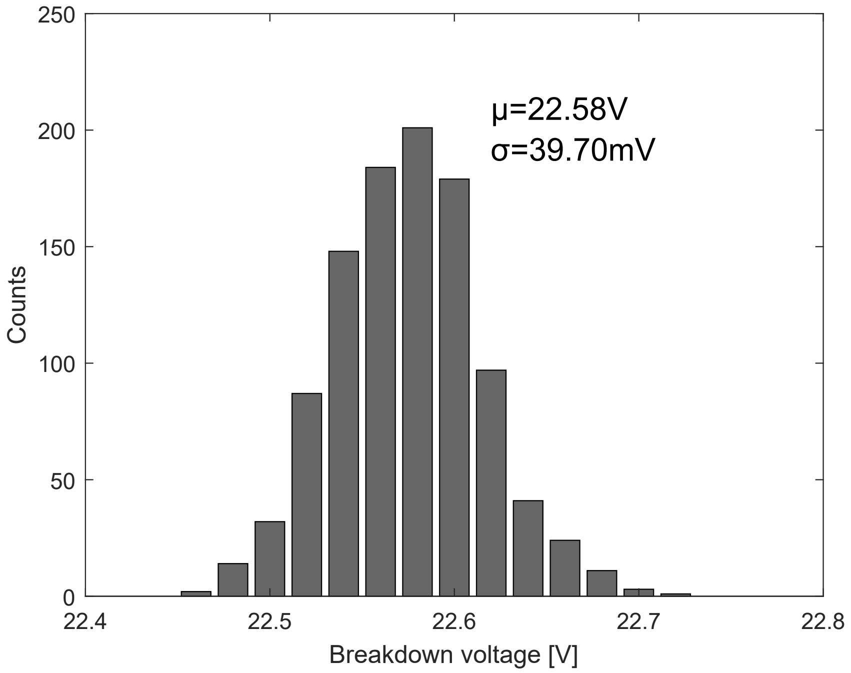


Figure 2.16: *kilo*Phase breakdown voltage measured by sweeping the SPAD bias voltage in 10 mV steps, acquiring dark frames and then determining when each pixel has fired.

2.5.3 Dark count rate

A custom cooling module was designed to be placed between the *kilo*Phase PCB and the motherboard. A 26 W 3 × 3 cm^2 thermoelectric cooler (TEC) was attached to a custom aluminium block that was in direct contact with the copper plane of the CoB PCB. An aluminium heat sink was attached to the hot side of the TEC and forced air flow was used to dissipate the heat into the environment. A Wavelength Electronics PTC2.5K-CH temperature controller [123] was used to drive the TEC and keep the temperature stable at the desired value using a thermistor attached to the copper plane of the PCB. Figure 2.17 shows the complete assembly.

The chip DCR was measured by accumulating 320000 8-bit dark frames with a 12.19 ns

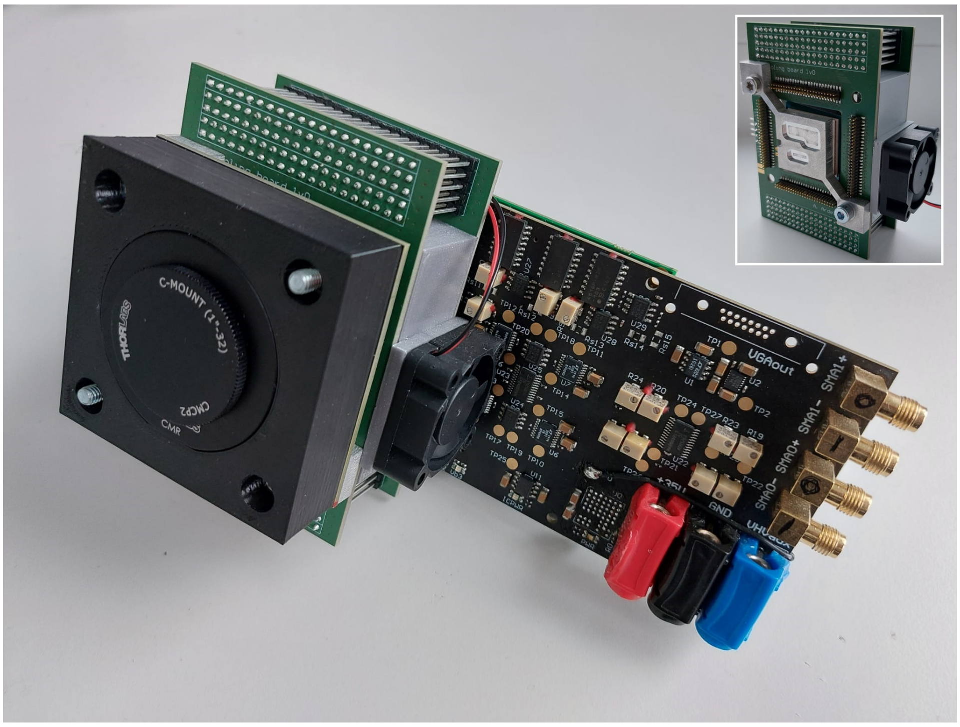
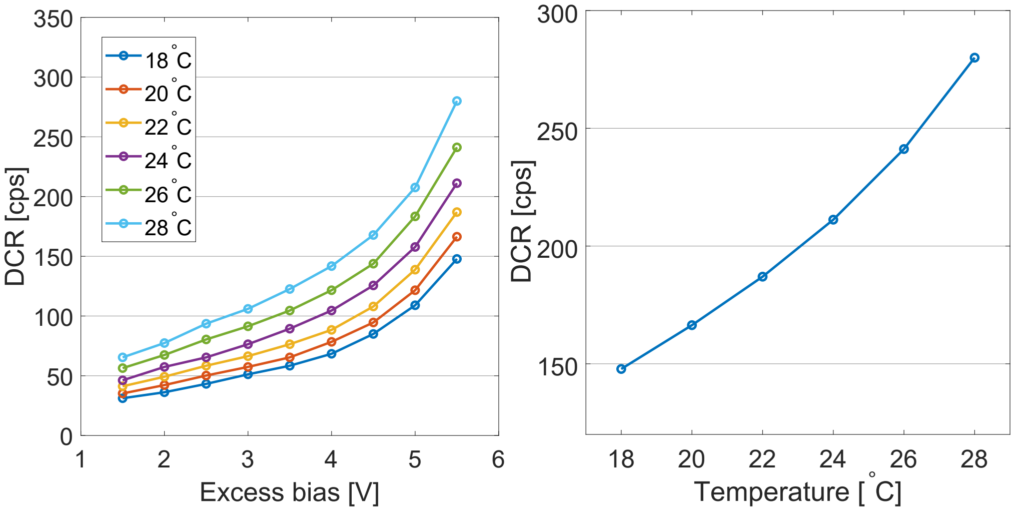


Figure 2.17: *kilo*Phase cooling assembly. The cooling module consists of two PCBs, a TEC, an aluminium block, a heat sink and a small fan placed on the side. The inset shows the module by itself, with the custom aluminium block designed to contact the CoB PCB visible in front.

gate for each binary frame, totaling an exposure time of $T_{EXP} = 0.995\,s$. The excess bias voltage of the SPAD was swept in steps of 0.5 V from 1.5 V to 5.5 V. The die temperature was stabilized within ±0.02 °C from five values between 18 °C and 28 °C, and checked with a FLIR ETS320 thermal imaging camera [124]. After each acquisition, the frames were summed together and the median value across the entire array was reported.

Figure 2.18a shows the median DCR as a function of SPAD excess bias voltage for multiple temperatures. The median value at 20 °C and 5.5 V excess bias was measured to be 166.4 cps. The dependency of DCR with temperature is shown in Figure 2.18b for an excess bias voltage of 5.5 V. An average increase in median DCR of 6.8%/°C was measured.

Figure 2.19 shows the DCR population distribution for multiple excess bias voltages. Under the same conditions as previously mentioned, the array exhibits a hot pixel percentage of 1.02%. The results are in accordance with the ones for the chip presented in [37] which has the same SPAD and array size. Changing the SPAD bias conditions does not significantly affect the percentage of hot pixels, indicating that their DCR values



(a)

(b)

Figure 2.18: (a) Median *kilo*Phase DCR as a function of excess bias voltage for temperatures between 18 °C and 28 °C. (b) Median *kilo*Phase DCR as a function of temperature for an excess bias voltage of 5.5 V.

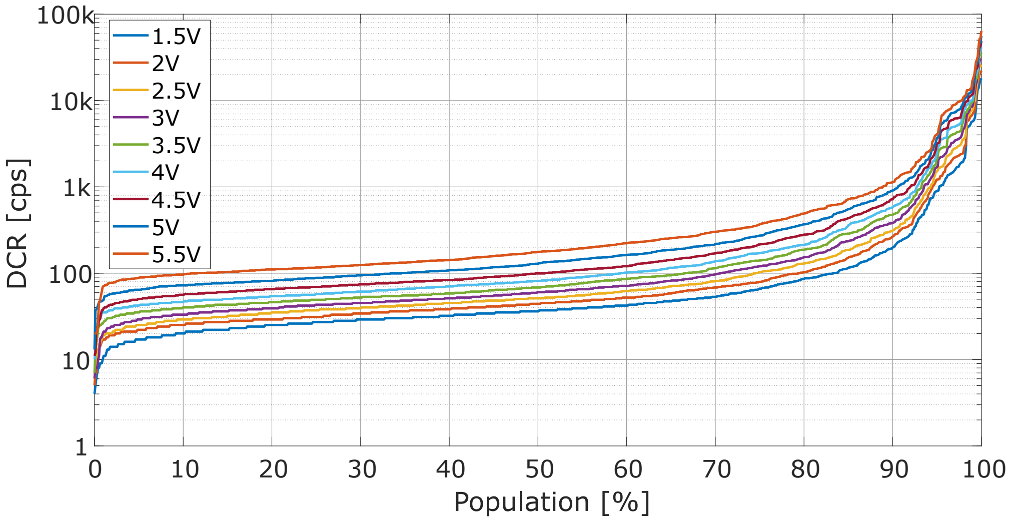
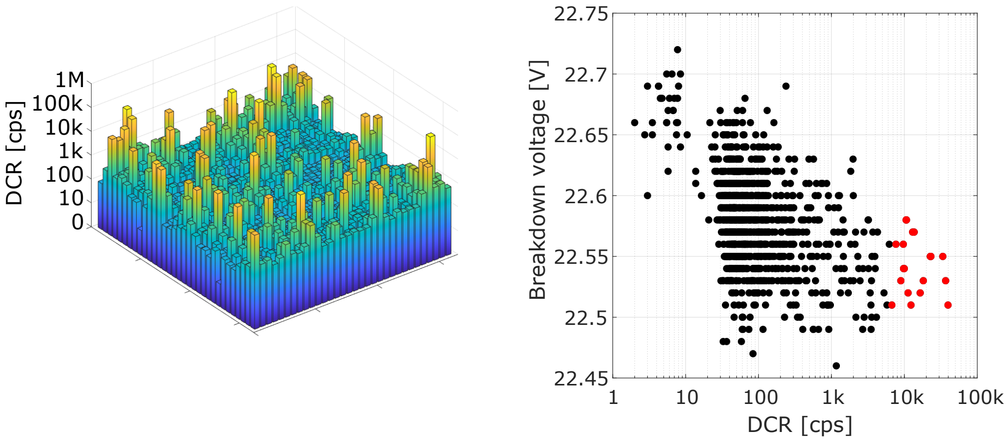


Figure 2.19: *kilo*Phase DCR population distribution for multiple excess bias voltages at 18 °C.

change at the same rate as the ones from the normal pixels. Figure 2.20b confirms that the hot pixels also have lower breakdown voltages compared to the rest.

Figure 2.20a is a 3D map of the DCR across the entire array at 20 °C and 5.5 V excess bias. A uniform spatial distribution of the hot pixels can be observed, similar to the one presented in [37].



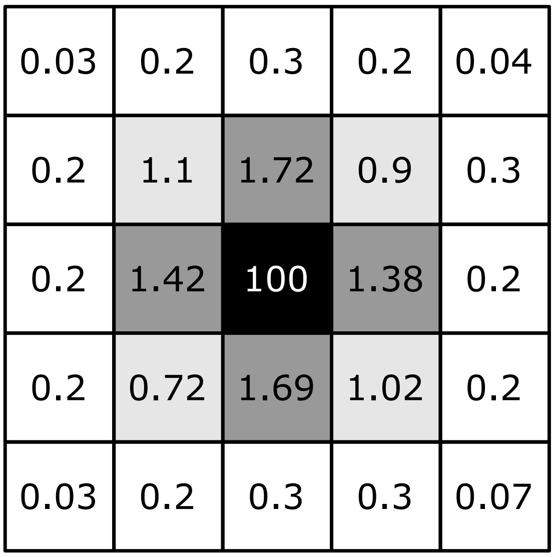
(a) (b)

Figure 2.20: (a) Median *kilo*Phase DCR across the entire array at 2.5 V excess bias and 18 °C. (b) Correlation between the DCR and breakdown voltage of each pixel under the same conditions. Hot pixels are shown in red.

2.5.4 Crosstalk

The DCR data was also used to measure the crosstalk between neighboring pixels, as described in Chapter 1.3.2. The 20 °C and 5.5 V excess bias voltage operating condition was chosen and a number of hot pixels were selected as reference. The selection criteria eliminated hot pixels with very high count rates so as to avoid extreme pile-up effects and at the same time only kept hot pixels without hot pixel neighbors. The median array DCR rate was subtracted from all the selected pixels and pile-up correction [104] was applied to estimate the true photon count. The crosstalk values for all the selected hot pixels were averaged to eliminate pixel-to-pixel DCR variations.

The crosstalk values are shown in Figure 2.21. The results were aggregated from 50 pixels across 8 samples. The average crosstalk was measured to be below 1.8% for the nearest pixels, and below 1.1% for the diagonal neighbors. The values suggest the existence of electrical crosstalk caused by routing the pixel signals in proximity to each other.



0.03	0.2	0.3	0.2	0.04
0.2	1.1	1.72	0.9	0.3
0.2	1.42	100	1.38	0.2
0.2	0.72	1.69	1.02	0.2
0.03	0.2	0.3	0.3	0.07

Figure 2.21: Average *kilo*Phase crosstalk at 5.5 V excess bias and 20 °C

2.5.5 PDP

The SPADs used in *kilo*Phase are exact copies of the ones from [37] and the same structure but with different dimensions was used in [71], [116] and [66]. PDP values for wavelengths between 400 nm and 850 nm at different excess bias voltages were already reported in [125] using a SPAD with an identical structure and a 11.7 µm active region diameter. Thus, no PDP measurements were performed with the 15.16 µm active region diameter *kilo*Phase SPADs as the same behaviour was expected. Figure 2.22 shows the aforementioned results.

2.5.6 Power consumption

The power consumption of the chip was measured during normal operation as a 10 bit imager. Data was read out of the chip at the end of every acquisition and the cycle was restarted. The computational units were configured to multiply the intensity data from each pixel with a random number. The chip was supplied at nominal voltage through four channels: 1.8 V for the core circuitry, 3.3 V for the pad ring and the clock trees used to distribute the gate and recharge signals and a high voltage channel for the SPAD. The average current was measured for each and the results are summarised in Table 2.1 and Figure 2.23.

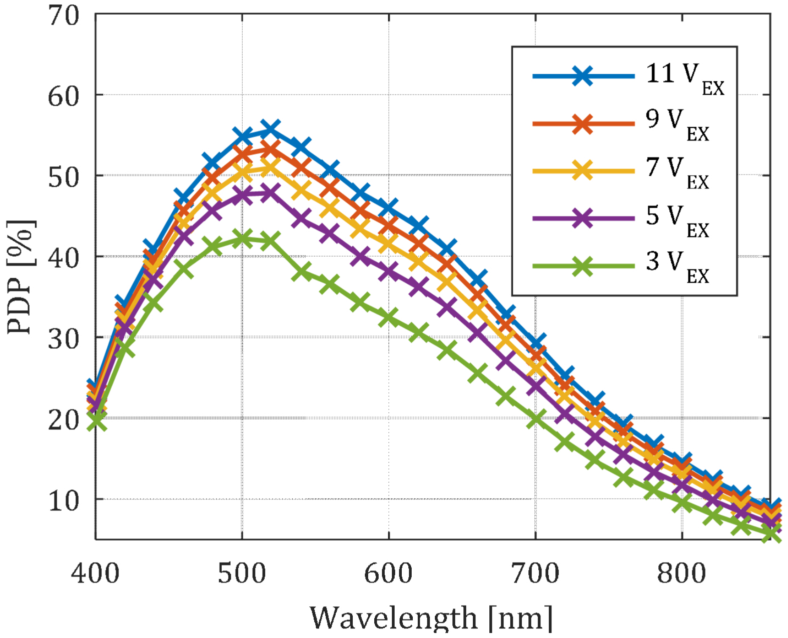


Figure 2.22: Photon detection probability reported in [125] for a smaller diameter but identical SPAD structure to the one used in *kilo*Phase.

Table 2.1: The power consumption of each voltage rail

Rail	**Power [mW]**
Pad ring	18.15
Clock trees	1.65
Digital cores	5.4
SPAD array	<13
TOTAL	<38.2

As expected, the majority of the power dissipation takes place in the pad ring because of the 3.3 V level and the large 75 bit width of the output bus. In future iterations of the chip, a more efficient serial interface can be implemented, preferably operating at lower voltage levels. The digital processing logic consumes less than 15% of the total power because of the system architecture and techniques such as clock gating that avoid power consumption in inactive circuit blocks. Little to no dependency on gate length or illumination conditions was observed.

2.5.7 Gate profile

The gate profile characteristics of a gated image sensor have a significant influence on the timing performance [84]. Significant effort was spent for a complete characteriza-

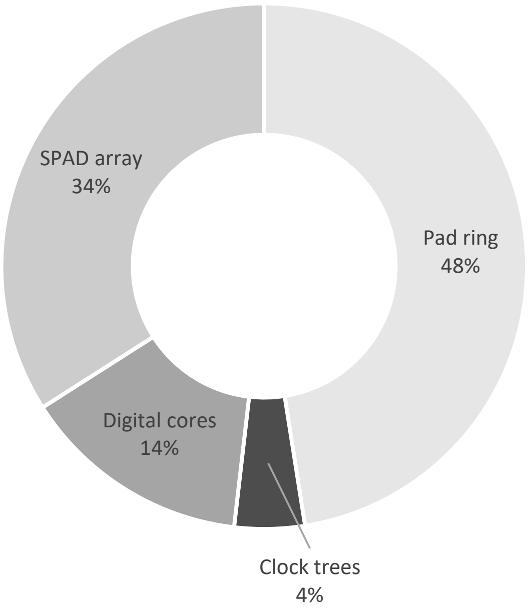


Figure 2.23: Average *kilo*Phase power consumption of every power rail.

tion of *kilo*Phase, similar to the one for other gated SPAD imagers such as [66], [71], [116].

The measurement setup was built around a Thorlabs Cerna single channel confocal microscope system [126] with a WFA2001 epi-illuminator module [127] and a 100 mm tube lens. A 517 nm pulsed laser [128] with an 80 ps pulse FWHM and timing jitter less than 3 ps was set to a 40 MHz repetition rate and used as an input for the illuminator module. An upside-down mirror was placed in the filter cube to redirect the laser pulse to the chip. The beam width was set using the microscope field stop to a size that covered the entire array. A 0.5 neutral density filter was used to reduce the laser power to a value that would not cause saturation of the pixels but would provide enough signal for an acceptable SNR.

The laser controller generated a 40 MHz trigger signal that was used to synchronize the system. The gate pulse width was kept fixed throughout the acquisition cycle but was shifted with 15.4 ps steps over multiple laser periods. At each step, 128 10-bit grayscale images were acquired and then the average value for each pixel was calculated. The data was used to determine the rise and fall times of the gate signal for each pixel, along with the skew across the entire array.

Figure 2.24 shows the histograms of gate rise and fall times. The average value of the former is larger than that of the latter because the rising edge is defined by both gate and recharge signals and is affected by the non-zero transition times of the signals, variations in transient times of the avalanche current and the probability of detecting photons during the recharge process [84]. Similar behaviour is seen in other SPAD imagers that have identical gating architectures [66], [71].

Figure 2.25 shows the histograms of rising and falling edge skews across the array, with insets representing color maps. All the values are relative to the pixel where the first edge was detected. The rising edge has a diagonal skew starting from the bottom left corner of the array caused by the orthogonal propagation of the gate and recharge signals from the bottom and left edges respectively. The vertical gradient of the falling edge is expected as it is the result of the propagation of a single electrical signal.

Figure 2.26a shows the median gate profile of the entire array for multiple gate width settings. The minimum achievable gate width was measured to be 4.38 ns, larger than expected. The reason was determined to be a combination of the undersizing of the input buffer of the gate signal and low signal integrity due to PCB routing. The latter will be improved in future versions of the board. The maximum gate width is firmware limited to 19.48 ns or approximately 80% of the laser repetition period. The variation across the array has a standard deviation of less than 38 ps. The associated histograms are shown in Figure 2.26b and 2.26c. The slight increase in counts immediately after the deassertion of the reset is attributed to afterpulsing from photons which arrive shortly before the application of the reset pulse. Residual charge from these events can persist inside the diode and be released at a later time. Increasing the duration of the reset pulse lowers the amplitude of the bump with an approximate reduction of a factor of 2 when changing the pulse duration from 5 ns to 10 ns.

2.5.8 Frame rate

The token-based pixel readout was validated by measuring the frame rate of the chip when operating as an intensity imager with various bit depths at reduced light levels. The aperture of a F1.4 8 mm lens was used to limit the average number of photons detected in a binary frame to approximately 7% of the number of pixels, resulting in an average of 6 detections per row. The binary depth of the intensity frame was varied between 1 and 10 bit and 128 frames were acquired at each point. The frame rate was measured by monitoring the frequency of a synchronization signal with an oscilloscope and finding the average over all 128 frames. The frame rate increase was reported relative to the frame rate obtained under the same conditions with the token readout disabled.

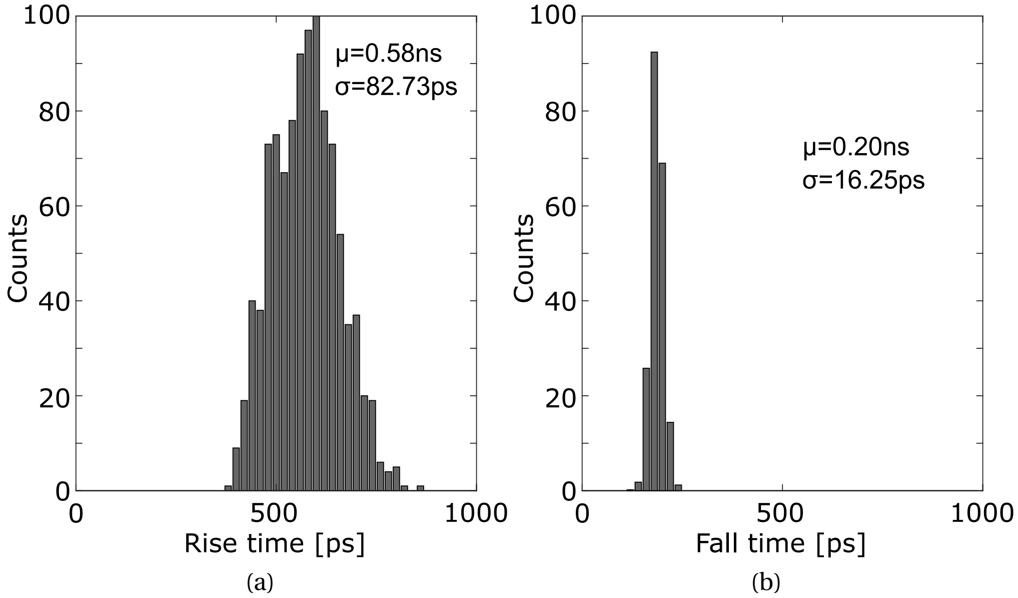


Figure 2.24: *kilo*Phase gate (a) rise and (b) fall time histograms. The average rise time is longer because the rising edge is defined by two signals and it also depends on the probability of detecting photons during the recharge process. Conversely, the falling edge is defined by a single electrical signal.

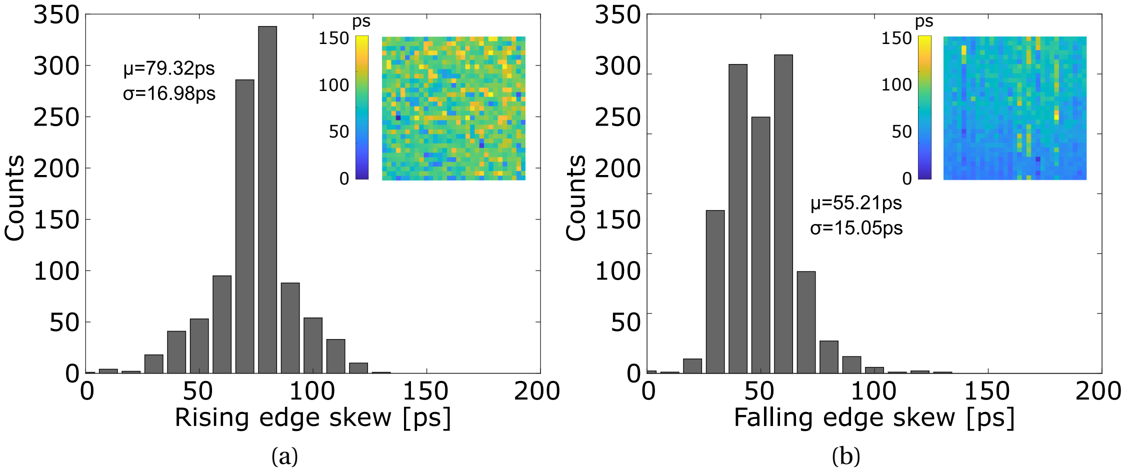
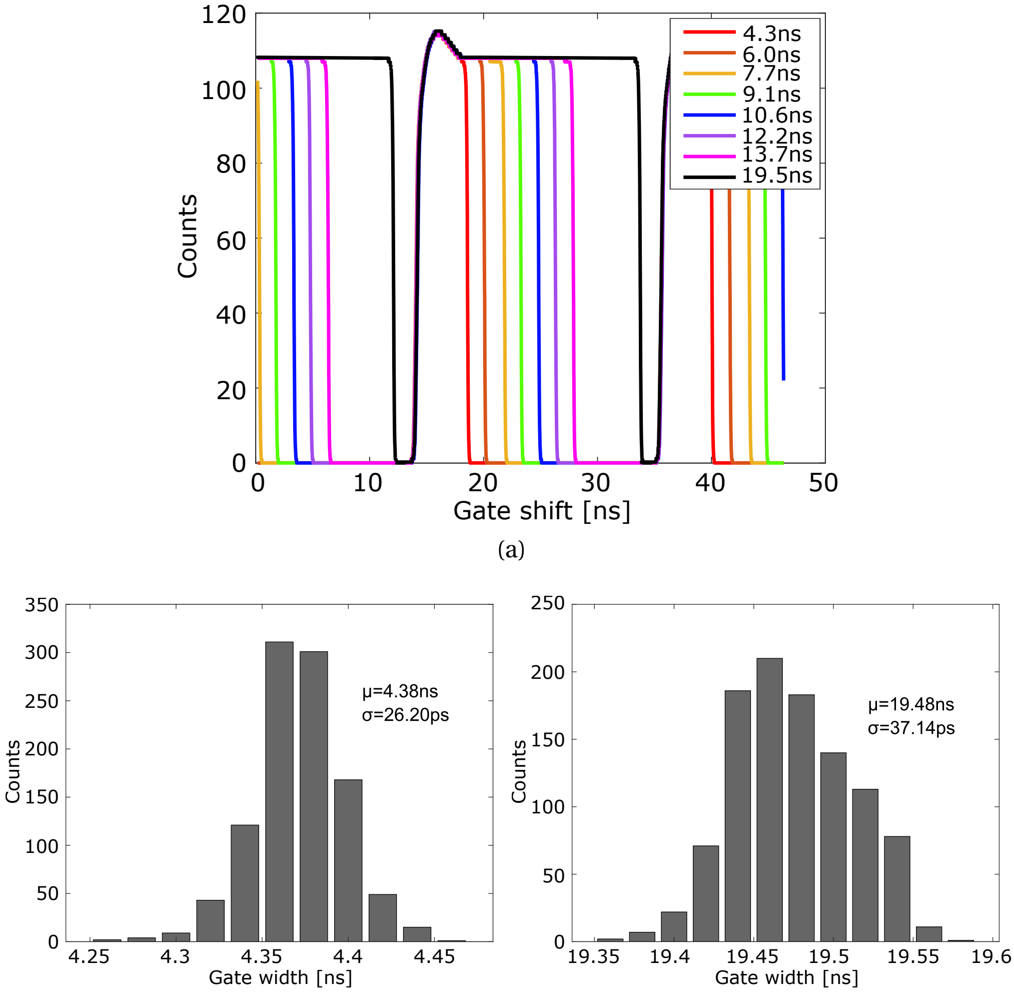


Figure 2.25: Histograms of *kilo*Phase (a) rising and (b) falling gate edge skews. The insets represent color maps of the values. A diagonal skew is observed on the rising edge due to the orthogonal propagation of the gate and recharge signals that form it. Only a vertical skew is present for the falling edge because it only depends on the propagation of the gate signal.



(a)

(b)

(c)

Figure 2.26: (a) *kilo*Phase gate width range. The slight increase in counts at the beginning of the gates is attributed to afterpulsing from photons which arrive shortly before the application of the SPAD recharge pulse before the gate. (b),(c) Histograms of gate widths across the entire array for the shortest (4.38 ns) and longest (19.48 ns) achievable configurations.

Figure 2.27 shows the average frame rate as a function of image bit depth. As expected, the frame rate doubles with every 1 bit reduction of the depth but saturates at the lower end of the range when the fixed chip readout time becomes the dominant factor. Maximum frame rates of 46340 fps and 227 fps were measured for 1 bit and 10 bit grayscale modes, respectively.

The inset represents the gain in frame rate when token-based pixel readout is enabled. A significant decrease is observed when operating at lower bit widths caused by the same fixed chip readout time. A maximum gain of 12% is obtained when operating in 10-bit mode. Future optimisations of the control firmware and an increase of the system clock frequency can be used to operate the system at higher gains. As described in Section 2.1 this gain is dependent on the illumination and is significant when operating in low light level conditions.

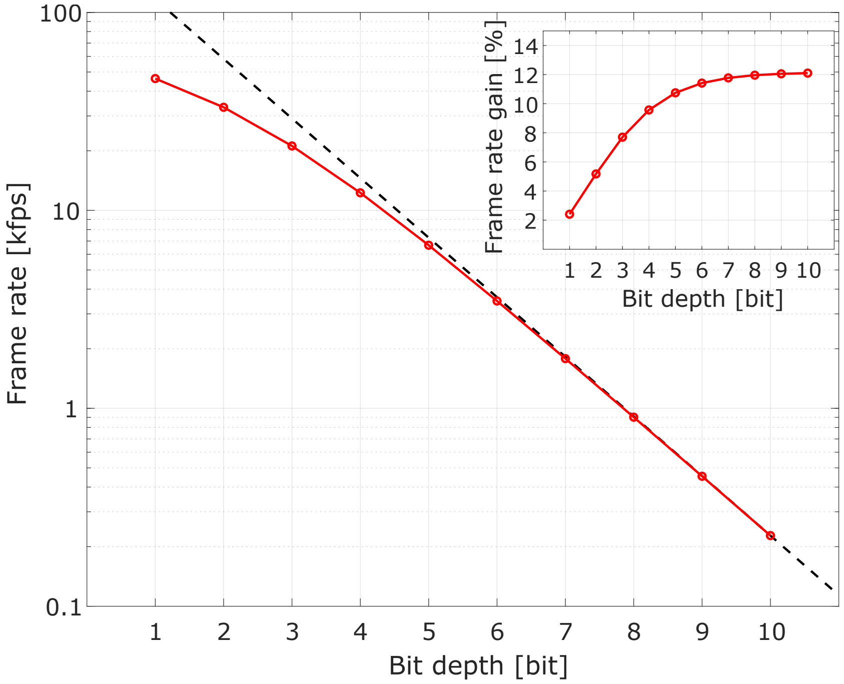


Figure 2.27: *kilo*Phase frame rate when operating as an intensity imager for multiple bit depths. As expected, the frame rate doubles with every reduction in bit depth and saturates at the lower end of the range when the fixed chip readout time becomes the dominant factor. The inset represents the relative frame rate increase obtained by using the token-based array readout. The effect of the fixed chip readout time is visible for low bit depths when the gain drops significantly.

2.5.9 Data rate

The chip output data rate was measured when operating as a 10 bit imager at 227 fps as in Chapter 2.5.8. Under these conditions, the data rate was measured to be 2.32 Mb/s, equivalent to 0.29 MB/s. A conventional binary SPAD imager would require an output data rate of 237.8 Mb/s to provide the same grayscale bit depth, indicating a 99% reduction for *kilo*Phase.

2.5.10 Microlenses

As discussed in Chapter 1.3, one of the main limitations of SPAD-based imagers is the decrease in PDE that comes from guard ring dimensions and minimum spacing rules in the technology. A widely used method for improving the PDE by increasing the fill factor is the use of microlenses deposited on each pixel in the array that concentrate the incoming light onto the pixel active area. Their performance is indicated by the *concentration factor* defined as [129]:

$$CF(\theta, \lambda) = \frac{C_{\mu l}(\theta, \lambda)}{C_{\overline{\mu l}}(\theta, \lambda)}, \tag{2.7}$$

where $C_{\mu l}$ and $C_{\overline{\mu l}}$ are the measured photon count rates with and without microlenses, λ is the light wavelength and θ is the angle of incidence. By multiplying the average concentration factor for all incidence angles in the system $CF_{avg}(\lambda)$ with the native fill factor FF of the array, we can obtain the effective fill factor:

$$FF_{\mu l}(\lambda) = FF \times CF_{avg}(\lambda). \tag{2.8}$$

Under ideal conditions, a convex lens can focus all collimated light at normal incidence ($\theta = 0^o$) into a single focal point and produce an effective fill factor of 100%. In practice, there is a minimum distance between adjacent microlenses which is a fabrication constraint that does not scale with the pixel pitch and which causes fill factor degradation. In addition, other fabrication non-idealities such as variations in microlense shape can reduce the expected CF [84].

As CF is heavily dependent on the angle of incidence θ, microlenses are designed for specific setups, in this case for microscopy. Due to the available space and the lack of need for a large angle of view, microscopy setups are telecentric lens systems with an average angle of incidence of $\theta = 0^o$ for all the pixels. This is because while the numerical aperture (NA) of the objective lens is usually very high in order to collect as much light as possible, the tube lens can have a low NA that results in a very small angle of incidence.

Figure 2.28 shows the low NA microlense array imprinted onto *kilo*Phase. The microlenses are exact copies of the ones for the detector presented in [37] to which I have made no contribution. Four pixels from the top row were left without microlenses so that they can be used to determine $C_{\overline{\mu l}}$ under the same conditions as $C_{\mu l}$. The distance from the bottom of the microlenses to the substrate surface is 65.8 µm and was chosen based on ray tracing simulations to maximise CF.

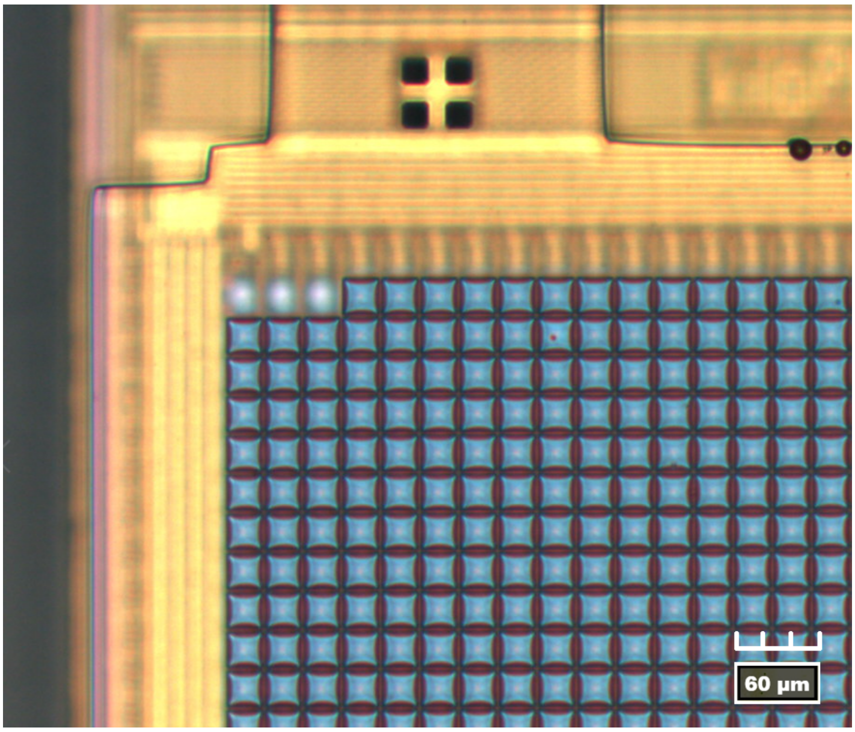


Figure 2.28: *kilo*Phase microlenses, detail of array corner. Three pixels without microlenses are visible on the top row.

The CF measurement setup was built around a Thorlabs Cerna single channel confocal microscope system [126] with a WFA2001 epi-illuminator module [127] and a 100 mm tube lens. A 517 nm pulsed laser [128] was used as an input for the illuminator module. An upside-down mirror was placed in the filter cube to redirect the laser pulse to the chip and the beam width was set using the microscope field stop to a size that covered the entire array. A 0.5 neutral density filter was used to reduce the laser power to a value that would not cause saturation of the pixels with microlenses and would provide enough signal for those without.

*kilo*Phase was operated asynchronously from the laser as a 10 bit intensity imager.

A total of 3000 images were acquired and CF was calculated for each microlensed pixel average value w.r.t. the average value of the 4 reference pixels. The resulting histogram and 2D map are shown in Figure 2.29. The average concentration factor was measured to be 2.44, 20% less than the expected value of 3.05, indicating slight discrepancies between the designed and fabricated lenses.

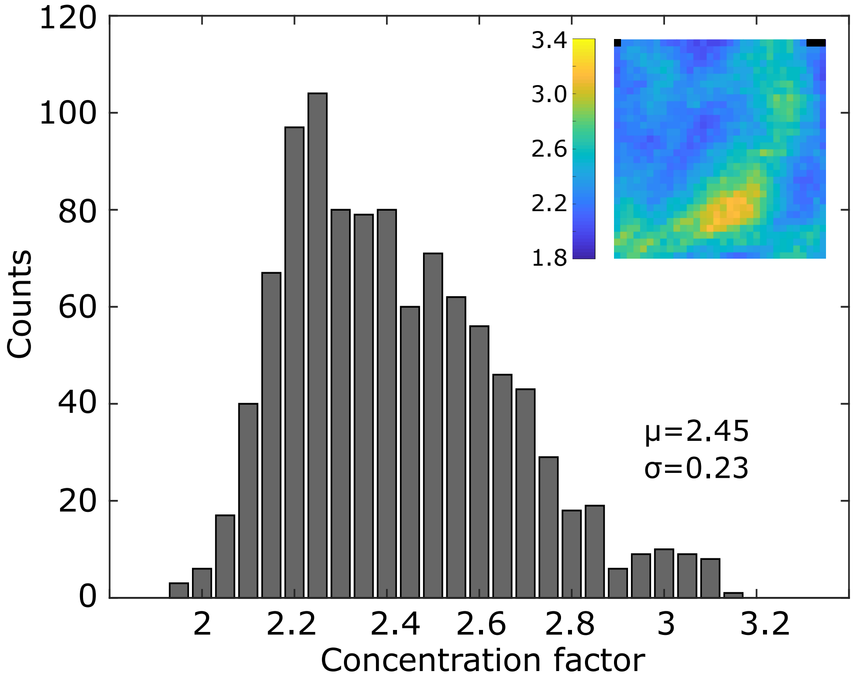


Figure 2.29: Measured *kilo*Phase microlense concentration factor histogram and corresponding 2D map.

2.6 Conclusion

*kilo*Phase, a 32 × 32 SPAD imager with vector processing capabilities was designed and characterised. The chip contains two 10 bit accumulator banks that can operate in a PING-PONG mode to eliminate system dead time by accumulating a new grayscale frame while processing the previous one. A group of 32 computational units can execute 2 MAC and one accumulation operation for each pixel, one column at a time. Two memory banks are used to store 128 multiplication coefficients and the 75 bit computational result for each pixel.

The system frame rate is increased by using a token-passing technique that only reads the pixels with detected photons. A 12% frame rate increase was measured when operating in 10 bit grayscale mode at 227 fps. Readout of the chip is only required at the end of the processing cycle when processed data is available for every pixel instead of raw pixel counts, reducing the output data rate of the system by 99% when operating in the same mode as previously mentioned.

A comparison between *kilo*Phase and other state-of-the-art SPAD imagers with computational capabilities is presented in Table 2.2. The power consumption is slightly better compared to the rest but it is also worth taking into account that *kilo*Phase was designed in an older technology node and is at a slight disadvantage. The output data rate is the key parameter at which *kilo*Phase excels. If normalized to the number of pixels, the data rate of [39] is better, however, currently *kilo*Phase has only been used as a 10 bit imager so the full advantage of the computational architecture has not been reported. When the FLIM firmware is complete, the chip will accumulate up to 128 10 bit images before readout is necessary, which will result in a drastic improvement in output data rate.

Table 2.2: *kilo*Phase state of the art comparison.

	***kilo*Phase**	**[37]**	**[63]**	**[130]**	**[64]**	**[39]**	**[65]**
Format	32 × 32	32 × 32	32 × 32	252 × 144	256 × 128	256 × 256	64 × 32
Pixel pitch	28.5 µm	28.5 µm	50 µm	28.5 µm	7 µm	9.18 µm	54/114 µm
Native fill factor	28%	28%	1.5%	28%	N/A	51%	16.3%
Technology	180 nm	180 nm	130 nm	180 nm	45/22 nm	90/40 nm	40 nm
DCR	0.54 cps/µm^2 @ 5 V	0.49 cps/µm^2 @ 5 V	0.59 cps/µm^2 @ 1.4 V	0.85 cps/µm^2 @ 5 V	0.34 cps/µm^2 @ 2.5 V	0.59 cps/µm^2 @ 1.5 V	N/A
Power	38 mW[a]	310 mW[b]	38.9 mW[c]	2.54 W	51.9 mW	77.6 mW	70 mW
Time resolving method	Gate	Column TDC	Pixel TDC	Column TDC	Pixel cluster TDC	Pixel cluster TDC	Pixel TDC
Timing resolution	4.18 ns (15.4 ps)[d]	48.8 ps	50 ps	48.8 ps	60 ps	35 ps	71 ps[e]
Operations	2× MAC and accumul.	Timestamp. or accumul.	Timestamp. or accumul.	Histogram or accumul.	Histogram and Coincidence	Histogram	Histogram
Data rate	12.53 Mb/s[a]	4.76 Gb/s	5.12 Gb/s	10.7 Gb/s	800 Mb/s	31.4 Mb/s	N/A

[a] Operating as a 10 bit imager with MACs active.

[b] Measured at a global throughput of 35.5 Mevents/s.

[c] Without ancillary and IO cells, measuring 10 ns intervals.

[d] Minimum gate shift.

[e] Operating at 14 GS/s.

3 *Mega*Phase

*Mega*Phase is a 1024 × 1024 SPAD imager developed in BSI 3D CMOS technology, with the top tier in 45 nm and the bottom tier in 22 nm. The pixel is built around an N+/PWELL SPAD at a 6.93 µm pitch. The chip contains 16384 computational units (see Figure 3.1) that each process 64 SPAD inputs and can perform two MAC and one accumulate operations per pixel. The 8 × 8 group of SPADs, referred to as a macropixel, assigned to each computational unit can be binned with multiple granularities to increase the effective dynamic range. All the computational units are independent from each other and can simultaneously function in different modes, resulting in a non-uniform imager that can have multiple regions with different spatial resolutions. The chip can be reconfigured at runtime, from one image acquisition to the next, with no dead time in between.

At the moment of writing this thesis, the *Mega*Phase fabrication process was completed. Initial tests have revealed an issue with two transistors in the pixel front end that were undersized because of inaccurate library models. A corrected layout was designed and will be fabricated in a future run. The chip functionality could only be partially verified and as a result, only simulation results will be presented in this chapter.

3.1 Motivation

*Mega*Phase was designed as an extension of *kilo*Phase taking advantage of the significant improvements that can be achieved by changing to a 3D integrated solution. The new technology allowed the design of a parallel and scalable architecture that operates at frame rates orders of magnitude higher than the previous implementation and at variable and non-uniform image resolutions that can optimize the output data

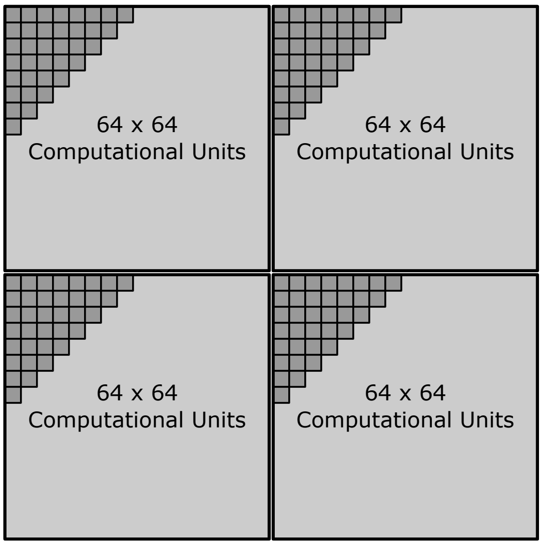


Figure 3.1: *Mega*Phase structure consisting of four quadrants of 4096 computational units each. The computational units are connected to groups of 8 × 8 SPADs and form a uniform pitch 1024 × 1024 pixel imager where each group can be independently processed either in intensity or computational mode.

rate.

The fundamental element of the *Mega*Phase architecture is the computational unit, a circuit block containing adders, a multiplier and RAM, that processes the outputs from 64 SPADs. At every processing iteration n, the computational unit can perform:

$$\begin{cases} D_G[n] = D_{PXL} \times \alpha_G + D_G[n-1] \\ D_S[n] = D_{PXL} \times \alpha_S + D_S[n-1] \\ D_I[n] = D_{PXL} + D_I[n-1], \end{cases} \tag{3.1}$$

where D_G and D_S are the real and imaginary parts of the complex pixel value, D_I is the integer intensity component, D_{PXL} is the integer result of accumulating a specific group of SPADs and α_G and α_S are the real and imaginary components of the complex coefficient. Depending on the operation mode, D_{PXL} can be computed using 64, 16, 4 or 1 SPAD outputs for every exposure, resulting in a gray pixel with a dynamic range of 6, 4, 2 and 1 bit respectively. As a consequence, the required exposure time T_{EXP} scales down exponentially:

$$T_{EXP}^{(b)} = \frac{T_{EXP}^{(1)}}{2^b}, \tag{3.2}$$

where $b > 1$ is the dynamic range of the pixel in bits and $T_{EXP}^{(1)}$ is the total exposure required for a conventional 1 bit SPAD pixel. The binning method is the same as the one used for processing the output data from quanta image sensors [131], [132], however, in that case there is no benefit in terms of speed or data rate.

Each computational unit functions independently from the rest and as a result, tiling multiple units to form a large format array does not increase the computation duration, the only frame rate limitation coming from the readout. In order to mitigate this, *Mega*Phase is read out using four 32 bit buses each assigned to a quadrant of the chip using a row and column addressing scheme that requires at most 8 clock cycles per computational unit. If a further increase in frame rate is desired, read out can be limited to a rectangular region of interest (ROI) centered in the middle of the array.

Finally, the independent nature of the computational units allows for multiple modes of operation to be present across the array at the same time. Various regions of different resolutions can be configured from one frame to the next which reduces the output data rate of the chip and as a consequence further increases the frame rate.

3.2 Processing flow

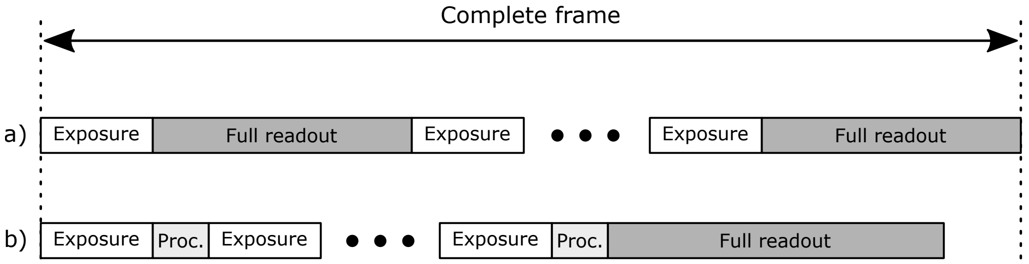


Figure 3.2: Comparison between the readout schemes of a conventional SPAD imager operating with a global shutter a) and *Mega*Phase b). In the latter case, every binary exposure is followed by a fast processing step. A full chip readout is only required at the end of the entire acquisition.

Normal sensor operation consists of three stages: exposure, computation and readout. The exposure begins with N_{EXP} openings of the gate signal that result in four possible images at the input of each computational unit: an 8×8 binary image if the 1 bit dynamic range is selected or a 4×4 or 2×2 or 1×1 grayscale image if the 2, 4, or 6 bit modes are selected. After exposure, the computation stage proceeds automatically and equations (3.1) are implemented simultaneously. Depending on the mode of operation, this step requires either 1 or 4 T_{CLK} clock cycles to complete. The exposure - computation sequence is then repeated as many times as desired. The operation terminates with chip readout which can take a maximum of 32768 T_{CLK} cycles to complete if the full array is read out at the most memory intensive operation modes. Figure 3.2 is a schematic representation of the operation of *Mega*Phase and a conventional SPAD binary imager operating with global shutter. Compared to the operation of *kilo*Phase described in Chapter 2.1, *Mega*Phase does not require any binary frame readout time due to the parallel architecture. The resulting time T_{frame} required to obtain either an intensity or computational frame with the entire pixel array is given by:

$$T_{frame} = N_{EXP} \times T_{EXP}^{(b)} + 32768 \times T_{CLK}, \tag{3.3}$$

where $T_{EXP}^{(b)}$ depends on the pixel dynamic range b, the desired output image bit format, the operating mode of *Mega*Phase and the scene exposure duration.

Table 3.1 summarizes the 6 operating modes of each *Mega*Phase computational unit and the corresponding full array resolution if all the units were identically configured. For specific modes, the output can consist of up to 16 intensity or 4 computational frames, but it is up to the user to decide on the exact number. Also depending on the mode, the format of an output frame can be 1 bit, 8 bit or 16 bit when operating in intensity mode, and 64 bit when in computational mode, consisting of two 24 bit complex and a 16 bit intensity component.

Assuming $T_{CLK} = 10$ ns and $N_{EXP} = 1$, the resulting frame rate for operating the entire *Mega*Phase array is presented together with the resulting speed-up. The former is obtained when compared with the frame rate of a 1024 × 1024 SPAD binary imager that can be read out in quadrants at 1 T_{CLK} per line. Under the same assumptions, the resulting data rate for operating the entire *Mega*Phase array is also reported in Table 3.1. The reduction in data rate is obtained when compared with a 1024 × 1024 SPAD binary imager operating at the same frame rate as *Mega*Phase.

3.3 Architecture

3.3.1 Pixel

Figure 3.3 represents the pixel schematic implemented in *Mega*Phase, with the corresponding layout shown in Figure 3.4. A total of 10 thin oxide MOS transistors and one thick oxide PMOS are used on the bottom tier to implement the pixel circuitry, while the N+/PWELL SPAD is placed on the top tier at a 6.93 µm pitch.

Table 3.1: *Mega*Phase frame rate and data rate compared to a theoretical 1024 × 1024 SPAD binary imager that can be read out in quadrants at 1 T_{CLK} per line. Both rates are computed assuming that the maximum RAM capacity is used in every configuration and that each exposure of a binary frame lasts one clock cycle.

	Resolution					
	1024 × 1024	512 × 512	256 × 256		128 × 128	
Type	Int.	Int.	Int.	Comp.	Int.	Comp.
Pixel dynamic range [bit]	1	2	4	4	6	6
Output frame format [bit/pixel]	1	8	8	64[a]	16	64[a]
Number of output frames	4	1	4	1	16	4
Maximum frame rate [fps]	12200	3000	12200	2000	32500	10800
Speed-up[b]	×0.06	×1	×1	×42	×170	×56
Data rate[MB/s]	1600	1600	1600	1070	1070	1430
Data rate reduction[c]	÷1	÷16	÷16	÷1024	÷4096	÷1024

[a] 24 bit for real and imaginary components, 16 bit for the intensity image.

[b] Speed-up is calculated by comparing the frame rate of *Mega*Phase with a conventional 1024 × 1024 SPAD binary imager that is read out in quadrants at a rate of 1 line per T_{CLK} cycle.

[c] The data rate reduction is calculated by comparing with the same binary imager assuming the two detectors operate at an identical frame rate.

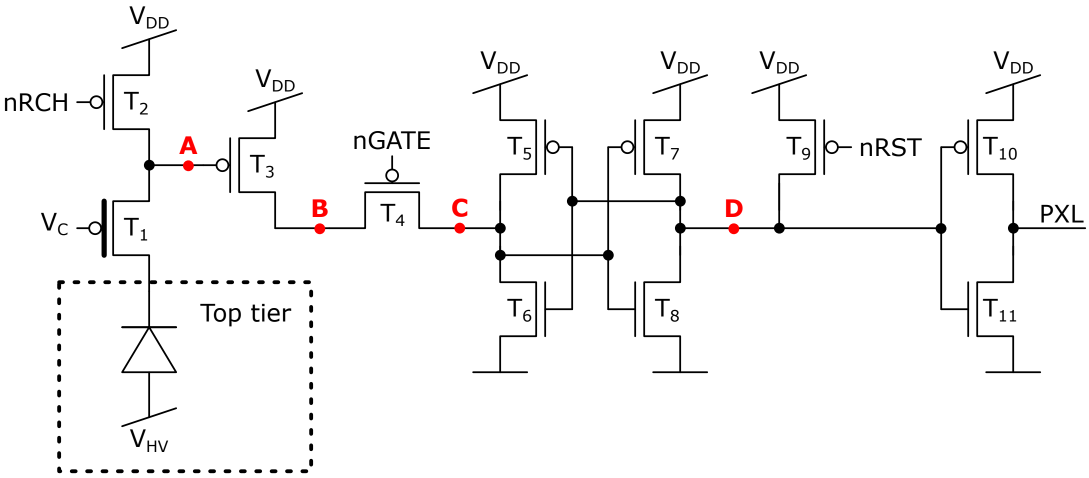


Figure 3.3: *Mega*Phase pixel schematic consisting of 7 PMOS and 3 NMOS thin oxide transistors along with a thick oxide PMOS cascode T_1. The N+/PWELL SPAD is implemented on the top tier wafer. The pixel has a clock-driven recharge (T_2) and a static memory cell consisting of two back to back inverters. The pixel is gated through transistor T_4.

Similarly to *kilo*Phase, cascode transistor T_1 and bias signal V_C are used to extend the bias voltage range of the pixel and T_2 implements a clock-driven recharge controlled by the $nRCH$ signal. When an avalanche occurs, the voltage at point A will drop, causing transistor T_3 to drive point B to V_{DD}. If the $nGATE$ signal is asserted, T_4 will short nodes B and C together, forcing the back to back inverter pair formed by T_5, T_6, T_7 and T_8 to toggle. Transistors T_{10} and T_{11} form an inverter that produces the output signal PXL based on the state of the memory cell. When the pixel is reset for a new exposure, T_9 controlled by $nRST$ will drive node D high and cause the memory cell to toggle back to the initial state before the avalanche. If during exposure an avalanche does not occur, node B will be floating as T_3 is not driven, and an opening of the gate will not cause a toggling of the memory cell.

3.3.2 Computational unit

The block diagram of a computational unit is shown in Figure 3.5. The 64 pixel outputs are connected to an adder tree that can be configured to sum only specific subgroups of its inputs, thus implementing pixel binning as shown in Figure 3.6. The output of the adder tree is processed by a group of two signed 24 bit multipliers and four signed 8 bit adders that can be reconfigured, depending on the mode, to simultaneously perform:

- two 24 bit MACs and one 16 bit accumulation

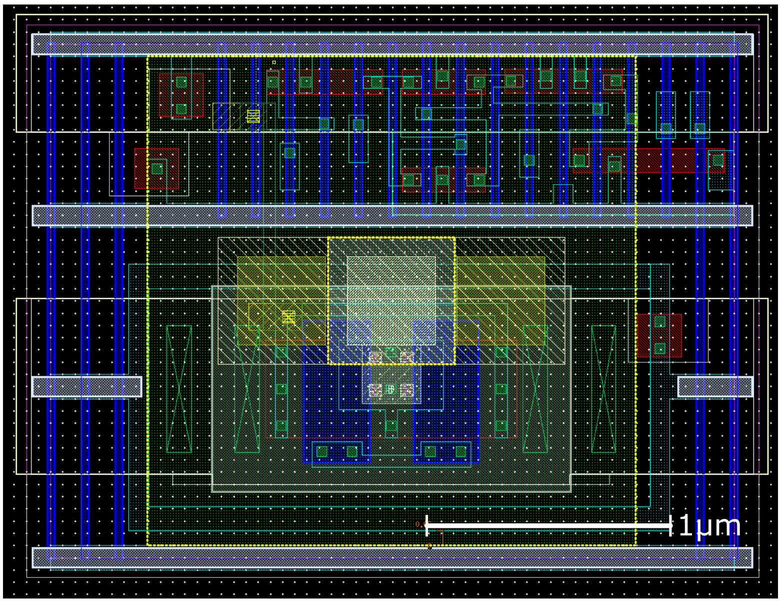


Figure 3.4: *Mega*Phase pixel layout. The thin oxide MOS transistors are all located in the upper section of the pixel. The large thick oxide PMOS occupies the entire middle and bottom sections. The pixel was designed to occupy 3 standard cell rows so that it can be used by the digital place-and-route tool without any issues. The large green square is a metal contact designed to connect to the top tier SPAD cathode.

- four 8 bit accumulations
- a single 16 bit accumulation

All the computation results are stored in a 256 bit RAM that also provides the additional operand for the MAC and accumulate operations. The structure of the RAM changes from 4 × 64 bit during normal operation to 8 × 32 bit for chip readout.

The 8 bit signed constants needed for MAC operations, as well as the configuration bits that determine the operational mode of the computational unit are stored in a 19 bit register. A three wire serial interface is used to load the register and then latch the current state. This behaviour allows the loading of new configurations and constants while the current frame is still being exposed/processed, which can result in no chip dead time.

3.3.3 Readout

As previously mentioned, the entire array is divided into four independent quadrants of 4096 computational units each. Chip readout is implemented simultaneously for

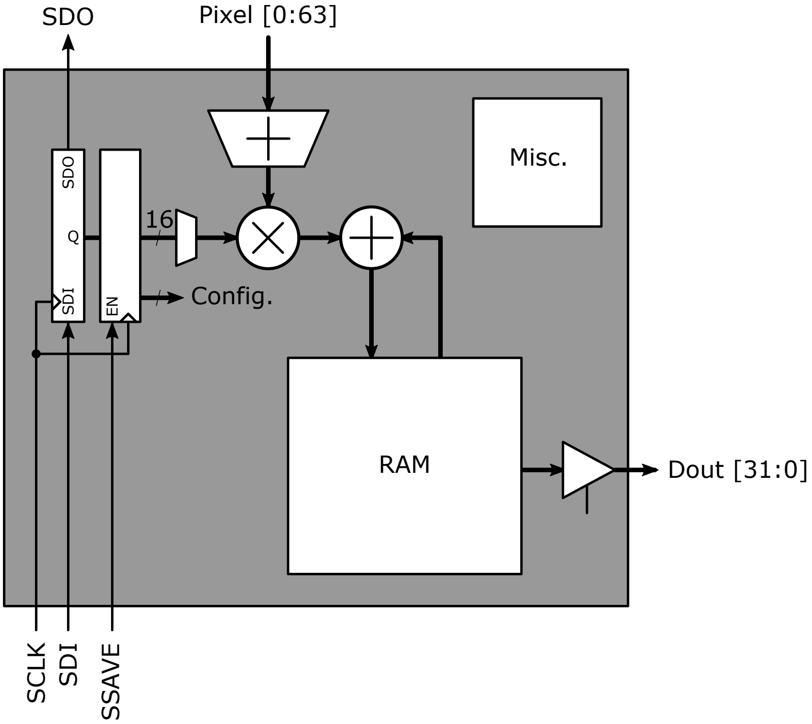


Figure 3.5: *Mega*Phase computational unit block diagram consisting of an adder tree that processes the 64 SPAD pixel outputs (Pixel[0:63]) in user selectable subgroups, signed integer MAC and accumulate units and a RAM block. Configuration data and mathematical constants are stored in a latched shift register that can receive new data using a three wire interface (SDI/SDO, SCLK, SSAVE) without interfering with current processing operations. Readout is performed using 32 bit output bus Dout. Additional miscellaneous circuitry is used for the configuration and control of the unit.

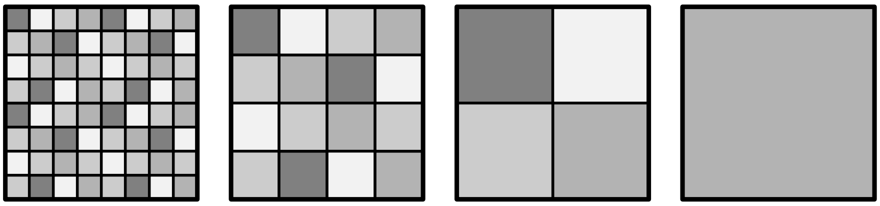


Figure 3.6: *Mega*Phase SPAD binning allows for larger dynamic ranges to be used at a cost of image resolution. Each computational unit can be configured to process 8 × 8, 4 × 4, 2 × 2 or 1 × 1 pixels as intensity or computational frames.

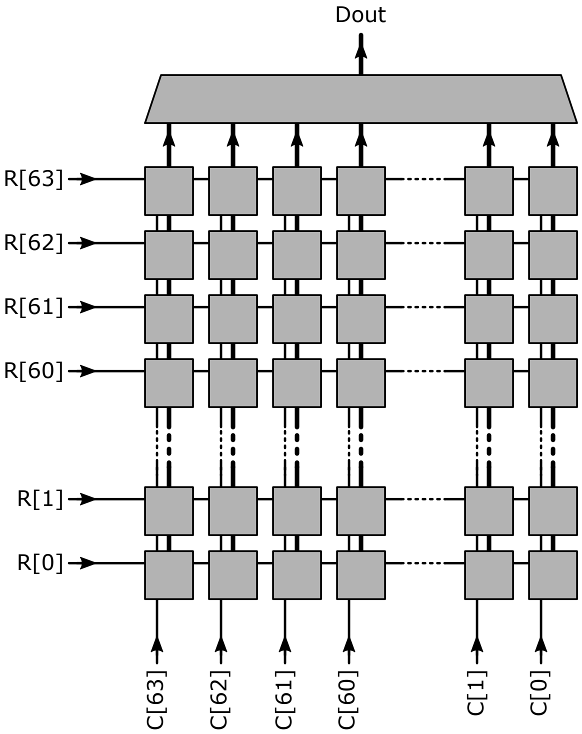


Figure 3.7: *Mega*Phase quadrant readout is implemented using a row and column addressing scheme. The order of the addresses is different for each quadrant so that the first unit is nearest to the center of the array. The schematic shown in this figure represents the upper left quadrant and has unit (0,0) in the lower left.

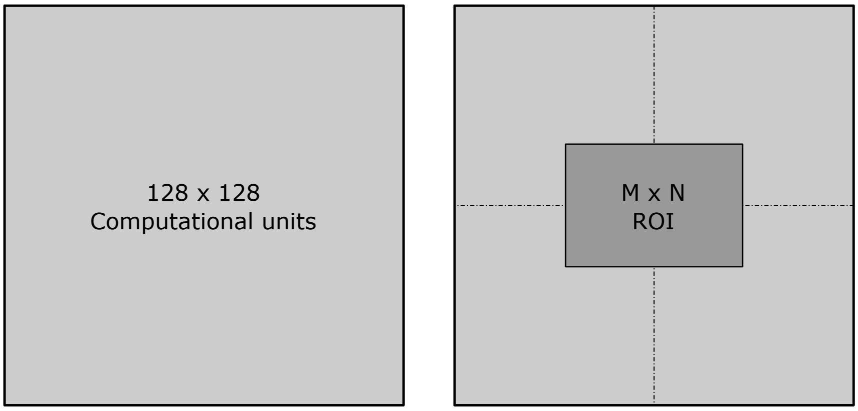


Figure 3.8: *Mega*Phase ROI readout can be used to increase the frame rate by limiting the operational section of the chip to a $M \times N$ rectangular group of computational units centered in the middle of the array.

each quadrant through a 32 bit parallel bus. A row and column addressing scheme is used to access the targeted computational unit from each quadrant as shown in Figure 3.7. The address bus is common to all four quadrants, but the order of the columns and/or rows is different; for example, the upper left quadrant has the (0,0) computational unit in the bottom right corner, whereas the upper right quadrant has it in the bottom left position. As a result, if only the first $\frac{M}{2}$ columns and $\frac{N}{2}$ rows are read out, a rectangular ROI of $M \times N$ computational units is formed in the center of the array as shown in Figure 3.8 which leads to an increase in frame rate.

3.4 Implementation

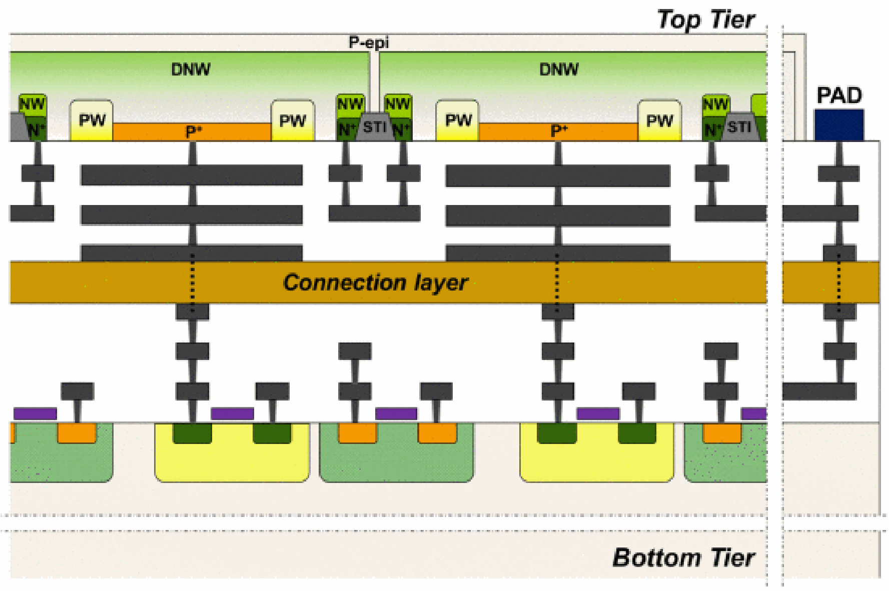


Figure 3.9: BSI 3D-stacked SPAD sensor structure similar to the one used for *Mega*Phase. Figure sourced from [133].

*Mega*Phase was designed as a 3D-stacked BSI imager, with hybrid bonds used to connect contacts on the top tier to corresponding exposed metal pads on the bottom tier [134]–[136]. Figure 3.9 shows a similar BSI 3D-stacked SPAD sensor structure presented in [133].

3.4.1 Top tier

The top tier was designed in a 45 nm technology node and consists of only an array of 1024 × 1024 SPADs. Each pixel is $6.93 \times 6.93\ \mu m^2$ with a drawn fill factor of 18% as shown in Figure 3.10. An array of metal contacts is distributed across the top tier and electrically connected to the SPAD cathodes, designed to be used for the 3D-stacking procedure.

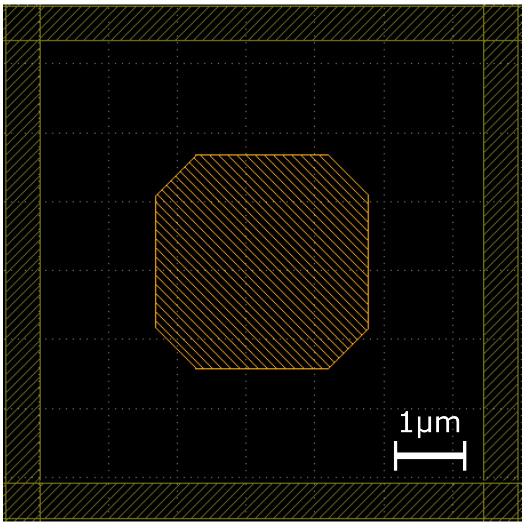


Figure 3.10: *Mega*Phase top tier pixel layout, showing the drawn SPAD active region in the middle and the metal routing used to connect to the anode. There is no additional circuitry on the top tier. The pixel is designed to be abutted to form the desired array size. The pitch is 6.93 µm

The SPAD structure was designed entirely by the foundry and appeared as a black box on our end. So far, its characteristics and performance have not been disclosed. The top tier layout consists of only 2 metals, the array of hybrid bonds and the I/O pads.

3.4.2 Bottom tier

The bottom tier was implemented in a hierarchical approach by firstly creating the processing core, then an array and finally the full design. The size and complexity of *Mega*Phase required the use of both manual and automatic techniques along with custom models developed for specific process steps. The completed design contains approximately 490 million transistors and the final design rule verification required 8 days of processor time to complete.

Design assembly

The pixel front end shown in Figure 3.4 was manually created using Cadence Virtuoso and was designed to occupy 3 standard cell rows so that it could be used by the digital place-and-route tools without issues. The transistors were sized using post-layout SPICE simulations where the SPAD was replaced by a model similar to the one presented in [120].

When the layout of the pixel front end was complete, an abstract .lef file representation was created using Cadence Abstract Generator along with a manually written .lib timing file. All the required delays and input loads were taken from the post-layout parasitic extraction. The pixel front end was then used as a standard component throughout the design.

The processing core was synthesized using Cadence Genus and the layout was created using Cadence Innovus. Figure 3.11 shows the *Mega*Phase processing core layout. Each core is 54.57 × 54.33 μm^2 and was designed to connect with its neighbor through abutment. A grid of 8 × 8 metal contacts are distributed across the layout and connect with the top tier SPAD cathodes via hybrid bonds. A combination of the top tier pixel pitch and the shrink factor of the 22 nm process resulted in an irrational pitch for the hybrid bonds on the bottom tier. In order to circumvent the issue, placement of the metal contacts is not uniform and has clusters of 2 × 2 elements as seen in Figure 3.11.

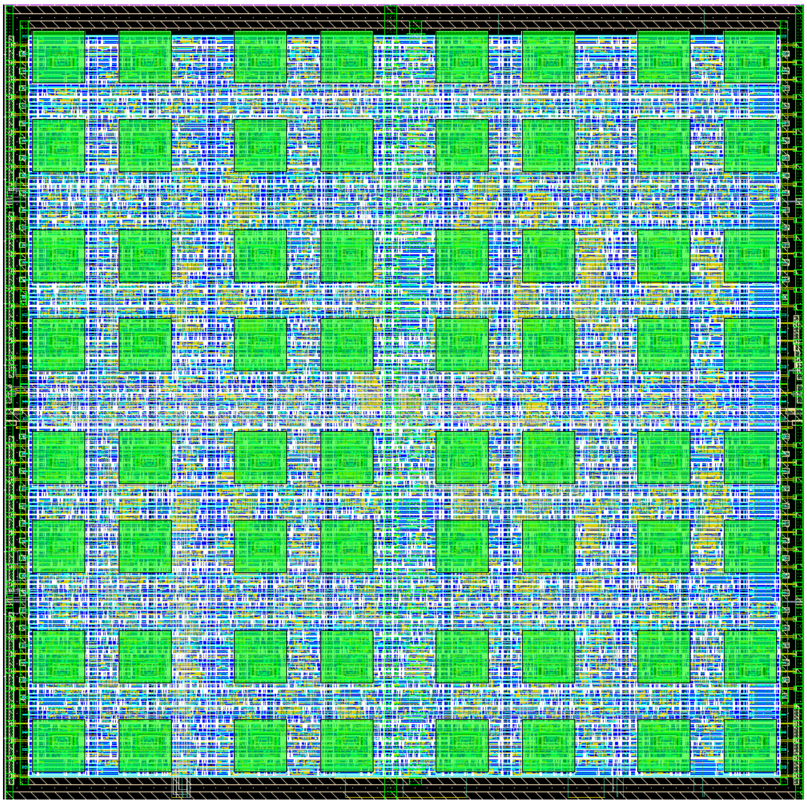


Figure 3.11: *Mega*Phase processing core layout. The 8 × 8 grid of metal pads for 3D integration is shown in green. The core is 54.57 × 54.33 μm^2 in size.

The layout for the processing core was limited to the first 5 metals. The core was exported as a hard macro and 16384 copies were tiled into the 128 × 128 array that forms *Mega*Phase. At this stage, the design became too large to be analyzed using the full .lib model exported by Innovus. Therefore, the model was overwritten by using directives that only specified the minimum information required for the timing analysis. The array was then exported as another hard macro.

The miscellaneous logic surrounding the array was fully synthesized and built using the same Cadence tools and then exported as a macro. The two parts were then combined and routed together using Cadence Innovus to form the *Mega*Phase core. Finally, the core was manually routed to the pad ring that contained only I/O cells as the pads were located on the top tier.

Gate and recharge distribution

Gate and recharge signal distribution across an array the size of *Mega*Phase had to be carefully designed to assure good signal integrity that preserved the shape of the gate and to minimize the propagation skew. A schematic of the final distribution network is shown in Figure 3.12 where three distinct sections are visible. Inside each processing core, the gate distribution was implemented by designing an H tree network using Cadence Innovus that minimized the skew and targeted a signal frequency of 125 MHz, corresponding to a minimum gate width of 4 ns.

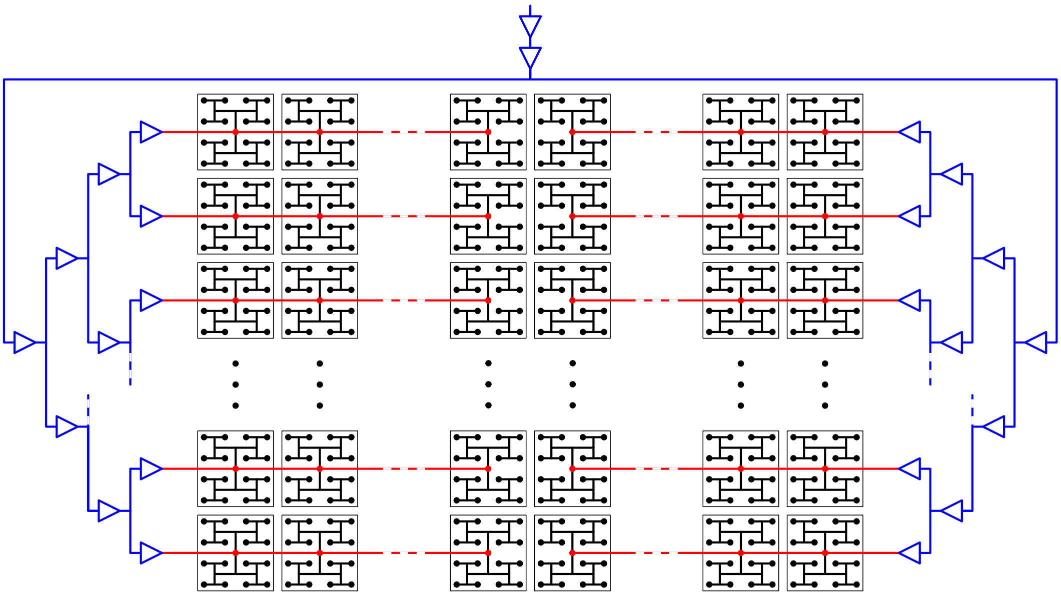


Figure 3.12: *Mega*Phase gate distribution network with the 3 stages shown in separate colors: top level tree (blue), propagation lines (red) and processing core H trees (black).

At the array level, the gate and recharge signals were propagated along dedicated lines for each row from two sides so that the horizontal skew between the processing cores was reduced. An H tree could not be used at this level due to the previously mentioned issues with modelling a design of this size. The simplified timing models were not good enough to be used for the generation of the tree using the design tools. All of the attempts to use slightly more complex models resulted in a complete crash of Cadence Innovus even when running on a machine with 675 GB of RAM and 64 Intel Xeon processing cores.

The propagation lines were designed with increased spacing from neighboring signals and had a larger width. Routing was kept on metal M6, above the processing cores, to ensure no obstacles were in the way. The signals were sandwiched between the M5 power distribution network of the cores and the M7 metal of the complete design power network. Both were routed as dense and uniform patterns of metal strips that preserved the uniformity of the parasitic elements attached to the propagation lines.

At the highest hierarchical level, the gate and recharge signals are distributed to the horizontal propagation lines by a tree that was synthesized and routed using the Cadence tools for the same target frequency as the H tree in the processing core. This ensured that the vertical skew was minimized and that the propagation to the two array halves was balanced.

Power distribution

Due to the hierarchical assembly method of the design, the power distribution network was also executed in multiple steps. At the processing core level, metal M5 was used to implement a dense pattern of vertical strips across the entire core that were periodically connected to the M1 metal of the standard cells to ensure uniform power distribution. The same technique was used for the digital logic surrounding the array where the gate and recharge distribution trees were located.

The chip power distribution network was created using the top 4 metals that are significantly thicker and wider than the rest. The M5 patterns from the processing cores and electronics were connected together with large and periodic metal strips in orthogonal directions to create a chip-wide grid. The edges of the grid were connected to a ring made with metals M9 and M10 that covered the entire perimeter of the chip. Finally, the ring was then connected to 8 pairs (VDD and GND) of pads distributed along all 4 edges of the chip.

MOS decoupling capacitors were placed in all the empty spaces in between the standard cells. The leakage current was reduced by more than one order of magnitude by replacing the foundry provided cells with custom-made ones where the transistors were replaced with their low power versions from a different library.

Clock distribution and timing

The system clock was synthesized for a maximum operating frequency of 100 MHz. The distribution network mirrored the one used for the gate and recharge signals. Timing closure was obtained in separate post-layout analyses for each block of the design. Due to the advanced technology node, a total of 8 library corners (summarized in Table 3.2) were used to take into account multiple operating conditions and process variations.

Table 3.2: All the library corners used for the *Mega*Phase timing analysis.

Corner	Transistor model	Voltage	Temperature	Interconnect
1	Slow-Slow	1.08 V	125 ^oC	RC-worst
2	Slow-Slow	1.08 V	0 ^oC	RC-worst
3	Fast-Fast	1.32 V	125 ^oC	RC-worst
4	Fast-Fast	1.32 V	125 ^oC	RC-best
5	Fast-Fast	1.32 V	0 ^oC	RC-worst
6	Fast-Fast	1.32 V	0 ^oC	RC-best
7	Typical-Typical	1.2 V	85 ^oC	RC-worst
8	Typical-Typical	1.2 V	85 ^oC	RC-best

Dummy generation

Dummy metal generation was performed in two steps. First, after the processing core layout was complete, dummy generation was run up until metal M5. This ensured that the timing performance of the design would not be affected by subsequent changes at the top level of the chip. The final dummy metal generation was then run on the assembled design for all metal layers.

Due to a technology specific issue, front-end-of-line (FEOL) dummy patterns were generated using a separate rule file which required the entire procedure to be done separately from the one for the back-end-of-line (BEOL). The same two step approach was used here as well, with the patterns for the processing cores being generated first, followed by a chip level generation.

3.5 Design verification

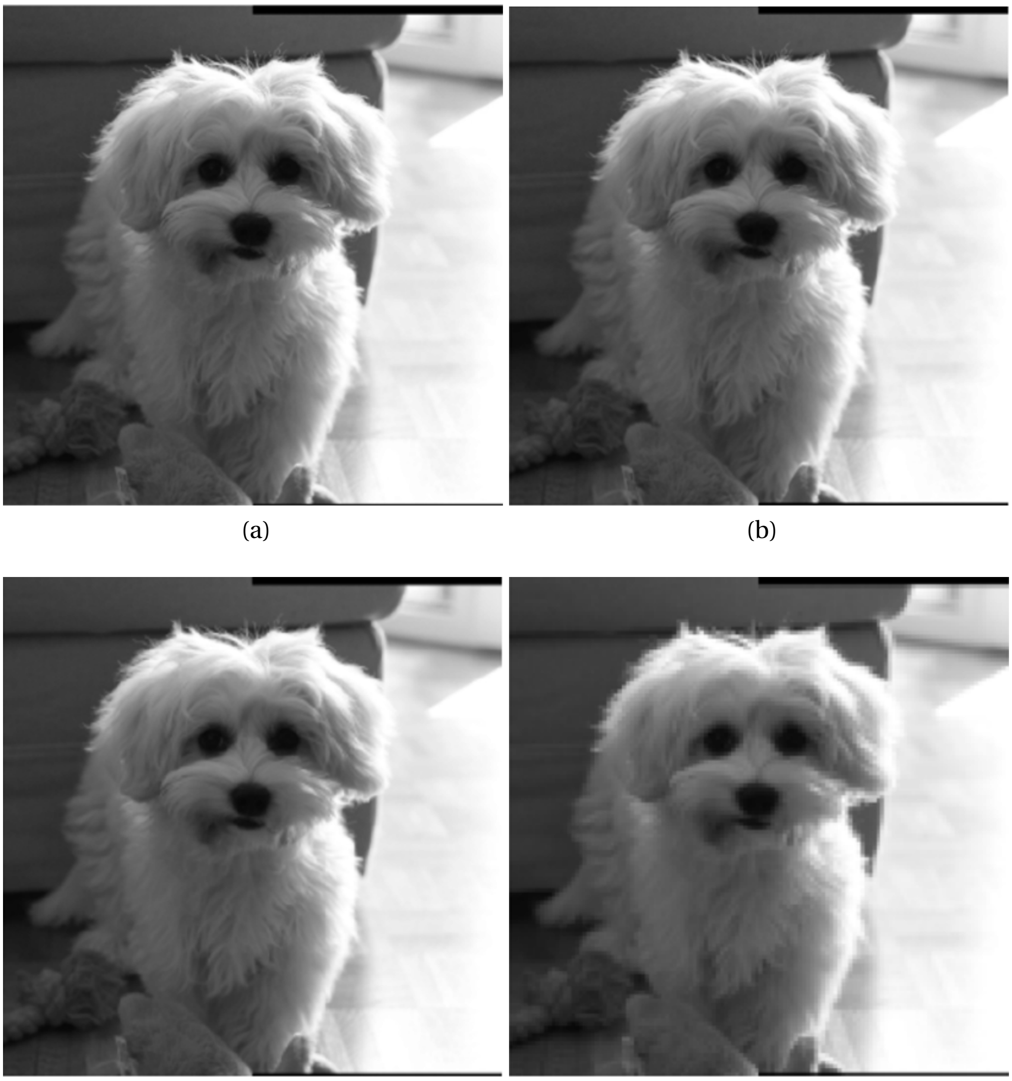
The architecture was validated throughout the design process, starting with behavioral simulations of the computational unit and ending with post layout simulations of the entire chip. Cadence Innovus was used to export the final Verilog netlist and .sdf file for corners 1, 4 and 7. The pixel front end was replaced by a custom Verilog model based on its standalone post layout SPICE simulations. Real and generated 8 bit grayscale images were used to simulate photon detections in the front end that would serve as inputs for the simulation model.

The intensity mode was tested for all 4 resolution modes described in Table 3.1 at the maximum frame rates. The results were then compared with the original 1024 × 1024 grayscale image and the binned versions of it for the other resolution modes. All output images were identical to the references. Figure 3.13 shows the 4 output images of different resolutions from one of the tests. The 1024 × 1024 output image was accumulated off-chip from the *Mega*Phase binary frames.

The computational mode was tested for both resolution modes described in Table 3.1 at the maximum frame rates. Random coefficient values were assigned to each core and the 3 components of the output D_G, D_S and D_I were compared with reference values computed from the same input image and coefficient combination. As expected, no discrepancies were observed in any of the tests. Figure 3.14 shows the results from one of the tests where a generated input image consisting of various geometrical patterns was used. The output resolution is 128 × 128. The 4 quadrants of the detector were configured to multiply the input with 4 complex numbers: $2-2j$, 0, $-3+3j$ and $-1+1j$, simulating a FLIM experiment with complex coefficients determined using equation (1.1). The chip was configured to only process the third output frame from the available four so that the memory access circuitry could be validated with the same test.

3.6 Conclusion

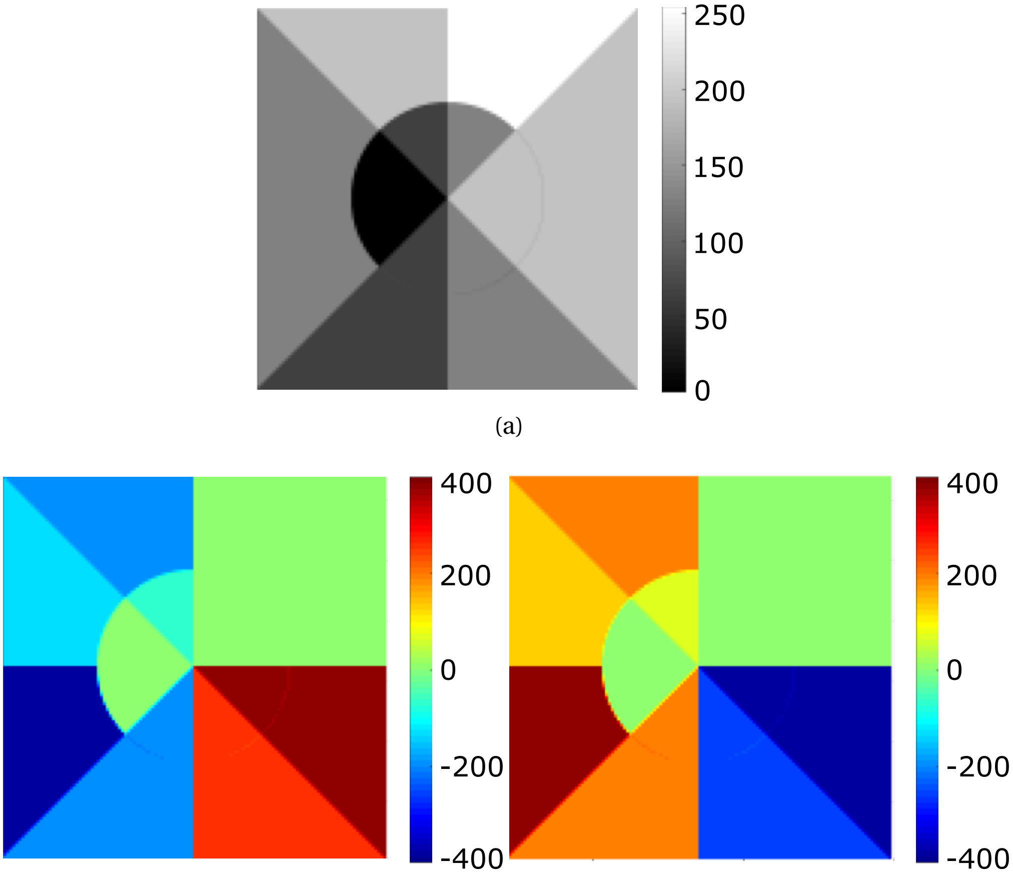
*Mega*Phase, a 1024 × 1024 SPAD imager with massively parallel processing capabilities was designed in a 45 nm/22 nm 3D stacked technology. The chip consists of an array of 128 × 128 processing cores, each capable of simultaneously processing 64 pixels by performing two MAC and one accumulate operations. The 64 pixels can be binned with multiple granularities to increase the dynamic range and reduce the required exposure time. Every processing core can operate independently, at different formats, resulting in a nonuniform imager that can optimize the output data rate. The chip can be reconfigured at runtime, form one image acquisition to the next, with no dead time in between. Depending on the mode of operation, the detector frame rate can be increased by a factor of 170 compared to a conventional SPAD imager of the same size and the output data rate can be as much as 4096 times smaller.



(a) (b)

(c) (d)

Figure 3.13: *Mega*Phase intensity mode simulation results: (a) 1024×1024; (b) 512×512; (c) 256 × 256; (d) 128 × 128. The lower resolution images were upscaled to 1024 × 1024 for display purposes. Image (a) was accumulated off-chip from the *Mega*Phase binary frames.



(a)

(b)

(c)

Figure 3.14: *Mega*Phase 128 × 128 computational mode simulation results: (a) intensity component D_I; (b,c) real and imaginary components D_G and D_S.

4 *Ultra*Phase

*Ultra*Phase is a 12 × 24 SPAD imager developed in 180 nm CMOS and 16 nm FinFET 3D stacked technologies. The top tier consists of 28.5 µm pitch front-side illuminated (FSI) pixels which have a native fill factor of 29.5 %. Every pixel output is connected to the bottom tier with an 8 µm through-silicon via (TSV). The bottom tier consists of a 3 × 6 array of independent processing cores that each have 4096 kbit of RAM, 6144 kbit of instruction memory, a fully reconfigurable front end, a timing module and a custom 32 bit CPU. The processing cores can share information with their direct neighbors and can synchronize with each other through the use of internal and external handshaking signals.

At the moment of writing this thesis, the *Ultra*Phase bottom tier has been successfully fabricated and tested. All the functionality was validated. The top tier has also been fabricated but the 3D stacking procedure is still taking place. All of the tests and applications presented in this chapter were performed using the bottom tier and bypass signals in place of the SPADs.

4.1 Motivation

*Ultra*Phase is a proof of concept reconfigurable and scalable computational imaging sensor. It was designed around a fully autonomous processing core, copies of which can be tiled to form large format imagers. The flexibility of the architecture stems not only from the ability to run custom programs in each core but also from the reconfigurable hardware at the pixel interface that can be customized through software at runtime.

The bottom tier contains only digital processing electronics and as a result the pixel front end needs to be implemented together with the SPAD on the top tier. This offers the possibility of using the bottom tier architecture as a generic readout IC coupled with custom detector technologies not limited to SPADs. However, the digital nature of SPAD pixels simplifies the interface because they can connect directly to the processing elements.

The bottom tier was designed for 3D integration performed by a third party, with large TSV landing sites similar in structure to traditional flip-chip ball bonding pads. This again adds a layer of customization when it comes to the choice of detector.

4.2 Architecture

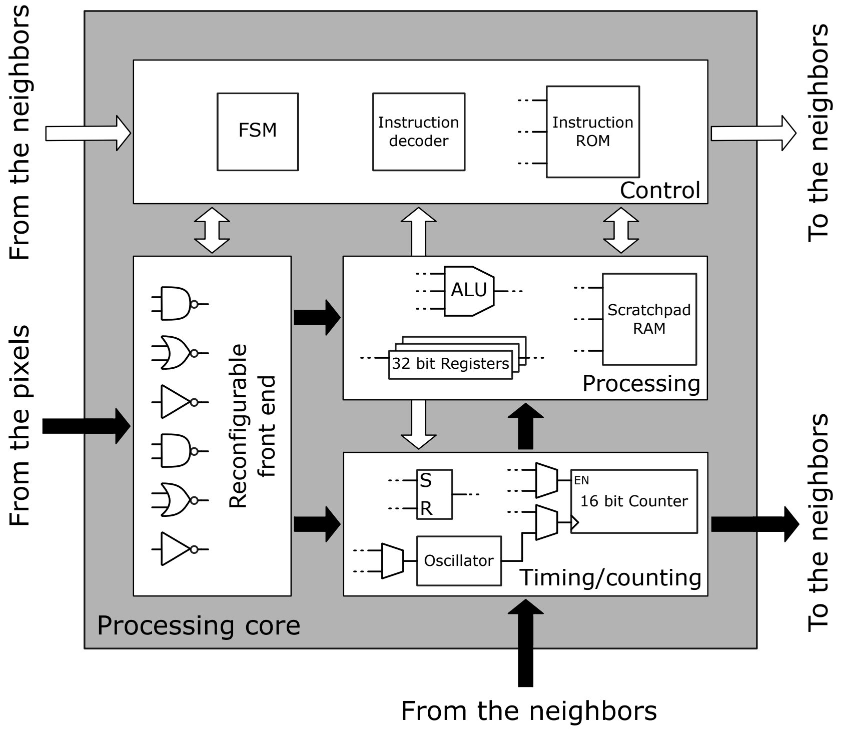


Figure 4.1: *Ultra*Phase processing core block diagram showing its four main modules. Input signals coming from 16 pixels connect to a *Reconfigurable combinational logic front end* that can preprocess the data before forwarding it to the *Timing* or *Processing* modules. The entire core is run by a *Control* module where the program is stored and decoded. Additional connections to neighboring cores can be used to transfer data, as timing signals or to synchronize multiple cores with each other or external signals.

Figure 4.1 shows the block diagram of a processing core. Each core is connected to 16 pixels on the top tier arranged in a 4 × 4 pattern. The input signals coming from the pixel circuitry on the top tier connect to a *Reconfigurable front end* circuit block that consists of combinational logic and lookup tables (LUT). This block can preprocess the input data before it reaches the *Timing* or *Processing* modules. The former is a specialized circuit that can perform various timing functions such as pulse width or phase shift measurements. The latter is a set of general purpose registers, an arithmetic and logic unit (ALU) and RAM. The entire operation of the processing core is coordinated by the *Control* module, where the instruction ROM and instruction decoder are located.

The Control module can receive inputs from neighboring processing cores and can in turn provide software controlled outputs used for synchronization. Similarly, the Timing module can receive inputs from the neighboring cores and can provide fast propagating signals through dedicated channels.

4.2.1 Reconfigurable front end

Figure 4.2 shows the schematic of the reconfigurable front end block. Sixteen pixel outputs are connected to a group of 4 LUTs (LUT0 to LUT3) in groups of 4 according to the diagram shown in figure 4.3. The purpose is to create binning of the 4 × 4 pixels into groups of 2 × 2. Each LUT can be programmed by the Control module using 4 instruction cycles to implement any logic function of the type:

$$Q_3Q_2Q_1Q_0 = a[0]\overline{P_3P_2P_1P_0} + a[1]\overline{P_3P_2P_1}P_0 + \ldots + a[15]P_3P_2P_1P_0 \tag{4.1}$$

where P and Q are the 4 bit LUT input and output and a is an array of 16 values of 0 and 1.

The outputs of the first layer of LUTs are connected to a secondary layer of circuits consisting of an adder and LUT4. The adder sums together the 16 bits of the LUT outputs into a single 5 bit number, essentially counting the number of 1s. LUT4 is a larger version of the other four, having an 8 bit input and a 1 bit output. Contrary to the adder, only two bits from the outputs of the previous LUTs are connected to it. LUT4 can also be configured by the Control module in 16 instruction cycles to implement any function of the type:

$$Q_0 = a[0]\overline{P_7P_6P_5P_4P_3P_2P_1P_0}+a[1]\overline{P_7P_6P_5P_4P_3P_2P_1}P_0+\ldots+a[255]P_7P_6P_5P_4P_3P_2P_1P_0 \tag{4.2}$$

where P is the 8 bit input, Q is the 1 bit output and a is an array of 256 values of 0 and 1.

The 16 pixel outputs also connect to two OR trees and can be individually masked using a set of AND gates. This secondary path is designed with a separate set of

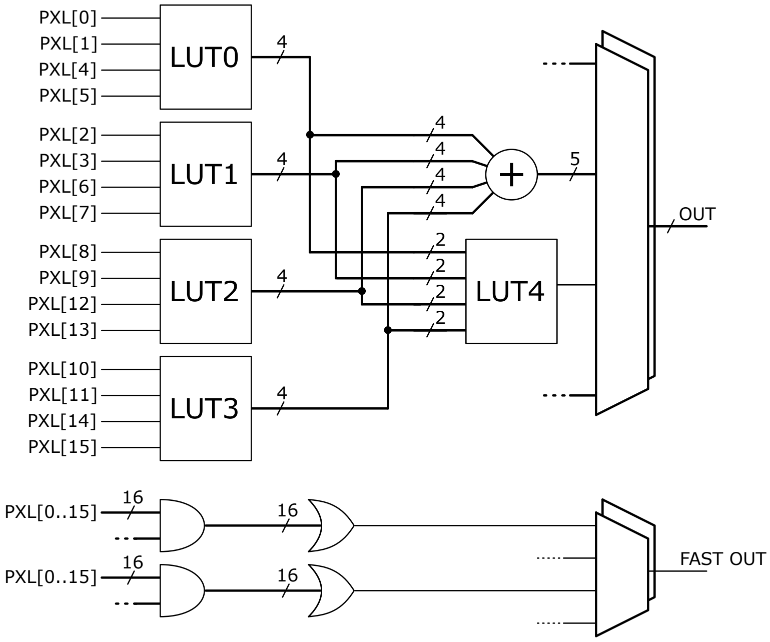


Figure 4.2: *Ultra*Phase processing core reconfigurable front end. The 16 pixel outputs are connected to multiple LUTs and an adder that combined can implement a wide range of combinational functions which can be updated at runtime through software. An additional path consisting of two OR trees with maskable inputs is designed to allow propagation of fast signals that can be used by the timing module or broadcast to the neighboring cores.

constraints to allow for fast signal propagation and can serve as input to the Timing module or the other neighboring cores.

Various signals from the front end block such as but not limited to the LUT, adder outputs and raw pixel values are connected to a DMUX. The Control module manages the DMUX which results in a software flexibility to select various pre-processed versions of the inputs without the need to reconfigure the front end.

4.2.2 Timing module

Figure 4.4 shows the schematic of the timing module implemented in each processing core. A 16 bit counter serves as the central element of the module, with its value used as the module output. A set of multiplexers are used to select which sources act as the counter clock and enable signals with a wide selection available for both cases. Table 4.1 summarizes some of the configuration options.

The enable signal can be sourced directly from the front end outputs or through

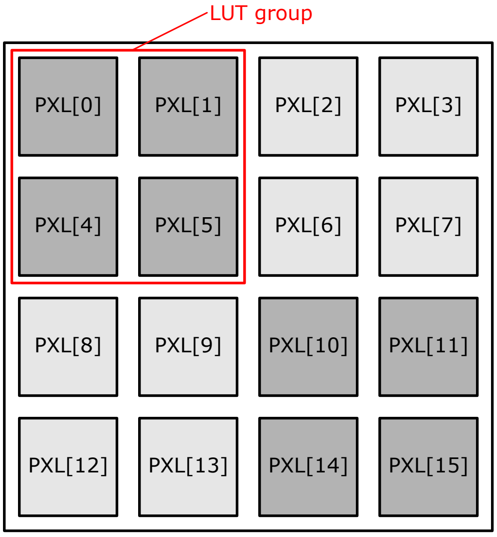


Figure 4.3: *Ultra*Phase processing core pixels are connected in groups of 4 to the reconfigurable front end to allow spatial binning.

a SR latch which can combine two separate input signals. In addition, pulses generated by neighboring cores or the Control module can also be used as clock or enable signals for the counter.

A local oscillator consisting of a ring of 7 NAND gates can be used to generate a higher frequency clock reference for the counter with the caveat that it is not PLL-stabilized and will present significant phase noise if the measurement interval is large.

The timing module generates two flags that can be used by the Control module for conditional instructions: a *counter overflow* and a *latch set*. The former is set when an overflow is detected in the counter and is essentially a latched 17th counter bit. The latter is the state of the input SR latch and can be used to detect the arrival of an input. The Control module is capable of reconfiguring the functionality of the timing module by setting all the multiplexers and resetting the counter and the two flags. A full reconfiguration requires two instruction cycles but for the majority of cases, a single cycle will suffice.

4.2.3 Processing module

As shown in figure 4.5 the Processing module consists of 6 general purpose 32 bit registers, a *byte selector* block, an ALU and RAM. The input of the processing module connects to the front end, the timing module and neighboring cores through a set of multiplexers managed by the control module. The output is the RAM memory itself

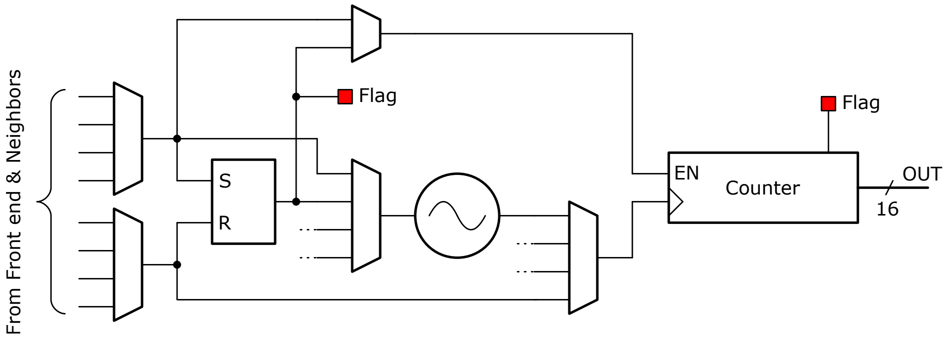


Figure 4.4: *Ultra*Phase Timing module schematic. The central element is a 16 bit counter that can use multiple inputs as either clock or enable signals. The counter output can be used locally or forwarded to the neighboring cores. The first overflow of the counter will set a flag that can be used by the control logic for conditional instructions. A local oscillator can provide a higher frequency module clock. The inputs coming from the front end or other cores can be used directly or passed through a SR latch that also serves as a flag signal for the control logic. The entire timing module can be reconfigured in a maximum of two instruction cycles at runtime.

Table 4.1: A subset of the possible Timer module configurations and resulting functions

Reference	Input A	Input B	Function
System clock	Front end	-	Coarse pulse width measurement
Local oscillator	Front end	-	Fine pulse width measurement
System clock	Front end	Front end	Coarse phase difference measurement
Local oscillator	Front end	Front end	Fine phase difference measurement
Software controlled signal	-	-	Pulse counter
Front end	-	-	Pulse counter
Neigbouring core	-	-	Pulse counter / Range extension

that can be read out by external system circuitry or a set of registers that connect to the neighboring cores.

The general purpose registers can be loaded with data coming from the input, the ALU or RAM. A special case is when the data is provided directly by the control module, when two separate instruction cycles are required: one for the lower and one for the upper 16 bits. The load signals for the registers are independently driven by the control module in order to enable writing of the same data into multiple locations simultaneously if required.

The byte selector is a specialized circuit used to shift or extract specific bytes from the 32 bit data word I present at its input. It can be used either by itself with a specialized

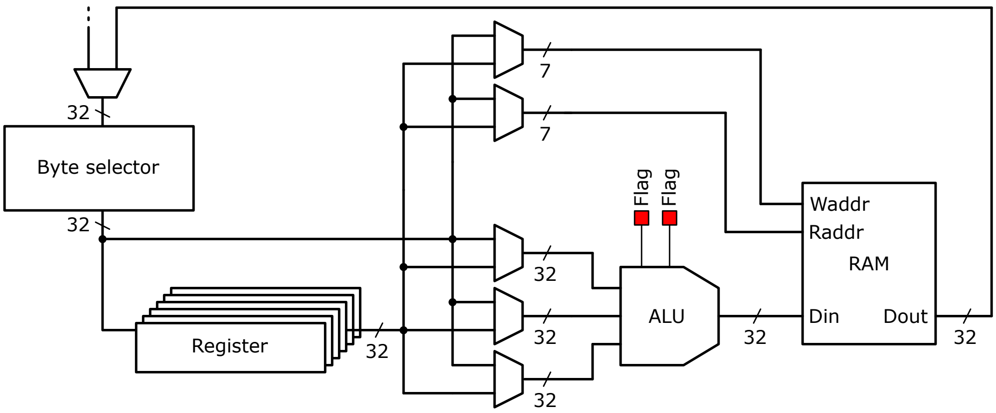


Figure 4.5: *Ultra*Phase Processing module schematic showing the 6 32 bit general purpose registers, the signed 32 bit ALU, the byte selector module and the scratchpad RAM. The byte selector is a specialized circuit used to shift or extract specific bytes from a data word, and can be used either by itself or during other instructions. A set of multiplexers changes the data path depending on the current instruction, while two flag bits set by the ALU can be used for conditional instructions ran by the control module. The RAM write and read addresses *Waddr* and *Raddr* are controlled independently which results in extra flexibility when choosing the data source and destination.

instruction or in combination with other operations. Table 4.2 summarizes the 8 manipulations that the byte selector can perform.

The ALU is a combinatorial circuit block with three 32 bit inputs and a single output of the same length. All the values are treated as signed integers for all the arithmetic operations. A 5 bit signal selects which one of the 25 possible operations is used to compute the output. Table 4.3 shows the 16 base operations of the ALU, omitting the 9 that are variations of ADD, SUB, MAC, SL and SR for clarity purposes.

Table 4.2: The 8 functions implemented by the Byte selector circuit

Function #	**Output**	**Effect**
0	I[31:0]	No operation
1	I[7:0]	Byte 0
2	I[15:8]	Byte 1
3	I[23:16]	Byte 2
4	I[31:24]	Byte 3
5	I[15:0]	Lower half
6	I[31:16]	Upper half
7	I[0:31]	Inverted bit order

Table 4.3: The 16 base operations performed by the ALU

Operation	**Result**
NOT	O = NOT Ain
AND	O = Ain AND Bin
OR	O = Ain OR Bin
XOR	O = Ain XOR Bin
NEG	O = - Cin
ADD	O = Ain + Cin
SUB	O = Ain - Cin
MUL	O = Ain × Bin
MAC	O = Ain × Bin + Cin
CMP	Ain < Bin ?
RL	O = Ain[30:0] & Ain[31]
RR	O = Ain[0] & Ain[31:1]
SL	O = Ain << 1
SR	O = Ain >> 1
MAX	O = max(Ain, Bin)
MIN	O = min(Ain, Bin)

In addition to the integer output, the ALU also generates two flags used for conditional jumps or instruction calls: the *Zero* and the *Carry*. Depending on the result of the arithmetic operation, these flags are either set or cleared and remain in the same state until another operation acts on them. As an exception, all of the logic operations reset the Carry flag.

The inputs to the ALU can be provided from multiple sources, either from the general purpose registers, an explicit RAM address, a pointer to a RAM address or a hard coded value in the instruction code. Similarly, the operation result can be written to a register or an explicit or pointed RAM location.

The memory is a dual port RAM block organized as 128 × 32 bit locations. The read and write address ports are independently controlled in order to allow the use of different data source and destination. In addition, the RAM can be accessed externally by overriding all the connections, a feature used for debugging and extracting the processor outputs.

4.2.4 Control module

The control module consists of an instruction memory, an instruction decoder circuit and finite state machine (FSM) logic. A 1 bit synchronization signal from each of the neighboring cores can be used at runtime for conditional instructions. Similarly, a 1 bit output signal is connected to each of the neighbors and can be either strobed or set through software. In addition, two 1 bit inputs and a 1 bit output that can be operated in the same fashion as the connections to the neighbors are present and are designed for synchronization with modules external to the design.

The instruction memory is a 256 × 24 bit dual port RAM block. In contrast to the scratchpad RAM from the processing module, only the read port of the instruction memory can be accessed by the processing core and as a result, from now on it will be referred to as a ROM. The write port is connected to a chip-level AXI bus and is only used during setup or in special cases where program execution is suspended and the instruction memory is rewritten at runtime.

*Ultra*Phase follows a *fetch* - *decode* - *execute* sequence that takes exactly 3 clock cycles for every instruction. During the *fetch* stage, the instruction pointed to by the program counter register (PC) is read from the ROM and passed to the instruction decoder, a combinatorial circuit that drives all the processing core control signals. At the *decode* stage, in addition to setting the control signals, data that is needed from the RAM is fetched, either by directly driving the RAM address bus or by using a general purpose register as a pointer. At the end of the final stage, the operation result is written to the requested destination and the PC is incremented.

Each instruction is 24 bits long and starts with a variable length opcode followed by the payload. Figure 4.6 shows three examples of instructions with their corresponding machine codes. The XOR instruction performs the operation on the data stored in general purpose registers $R0$ and $R1$ and writes the result to register $R2$. In this case, the payload consists of the 3 bit addresses for the registers. The multiplication (MUL) instruction uses register $R4$ for the first operand, an explicit value for the second and stores the result back into $R4$. The machine code consists of a 5 bit opcode, the 3 bit register address and a 16 bit representation of the second operand. Finally, the return from subroutine (RET) instruction is showed as a special case where the opcode is the only information encoded and the payload is ignored.

Due to the lack of available silicon area, instruction pipelining which would require more advanced techniques such as operand forwarding, out-of-order execution or branch prediction has not been implemented [137]–[145]. The PC always starts from 0x00 after a system reset and will loop back to this value if the program is not properly written and an increment beyond 0xFF is encountered.

A call stack is used to store information about active subroutines of the program.

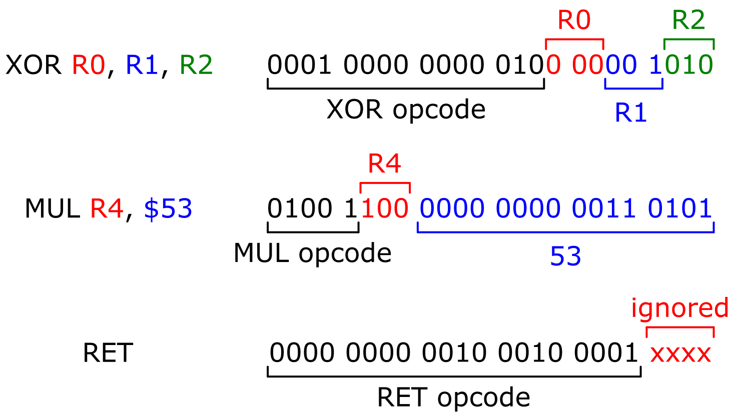


Figure 4.6: *Ultra*Phase instruction machine code. Three different instruction types are presented to showcase the variable length of the opcode and the multiple uses of the payload.

Contrary to conventional CPU architectures, the stack is implemented as a standalone register bank with automatic push and pop functionality instead of using part of the data RAM [146]. Therefore, there is no need for a stack pointer register or stack management in the program and there is no risk of corrupting the execution by overwriting data. The small depth of the stack limits the number of nested subroutine calls to 4, with the return information for any further calls being lost.

4.2.5 Processor array

*Ultra*Phase comprises 18 processing cores arranged in a 6 × 3 grid as shown in Figure 4.7. Each core is connected to its four direct neighbors through a 32 bit bidirectional data bus and a 1 bit bidirectional timing signal. In case one or more neighbors are missing, the corresponding signals are connected to a register.

Programming and readout is performed through a conventional AXI bus, each core being mapped to 400 memory locations which include the instruction ROM, RAM and any accompanying registers when required.

Two input signals are distributed to all 24 cores and can be used for synchronization. An output signal from the second core on the second row is directly connected to an I/O pad, while the corresponding ones from the rest of the cores are read through a register. In addition, this core has all 16 pixel inputs connected to I/O pads for debugging and integration with other detectors.

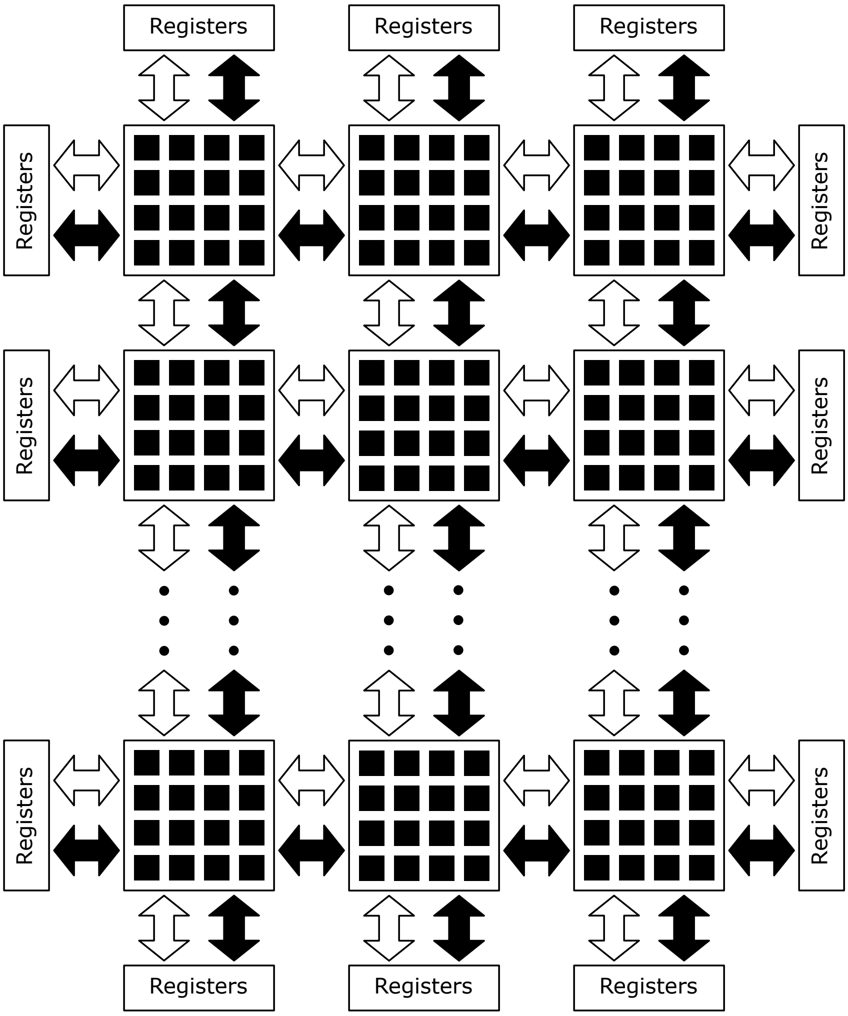


Figure 4.7: *Ultra*Phase 6 × 3 array of interconnected processing cores. In case one or more neighbors are missing, the corresponding signals are connected to registers that can be accessed via the SoC AXI bus. Both data (black arrows) and synchronization signals (white arrows) can be shared between direct neighbors. The global synchronization signals and the external connections to the second core of the second row are omitted from the diagram for clarity.

4.3 Implementation

*Ultra*Phase was designed as a 3D-stacked FSI imager, with the top tier smaller than the bottom one to allow access to I/O pads. Figure 4.8 is a representation of the assembled chip.

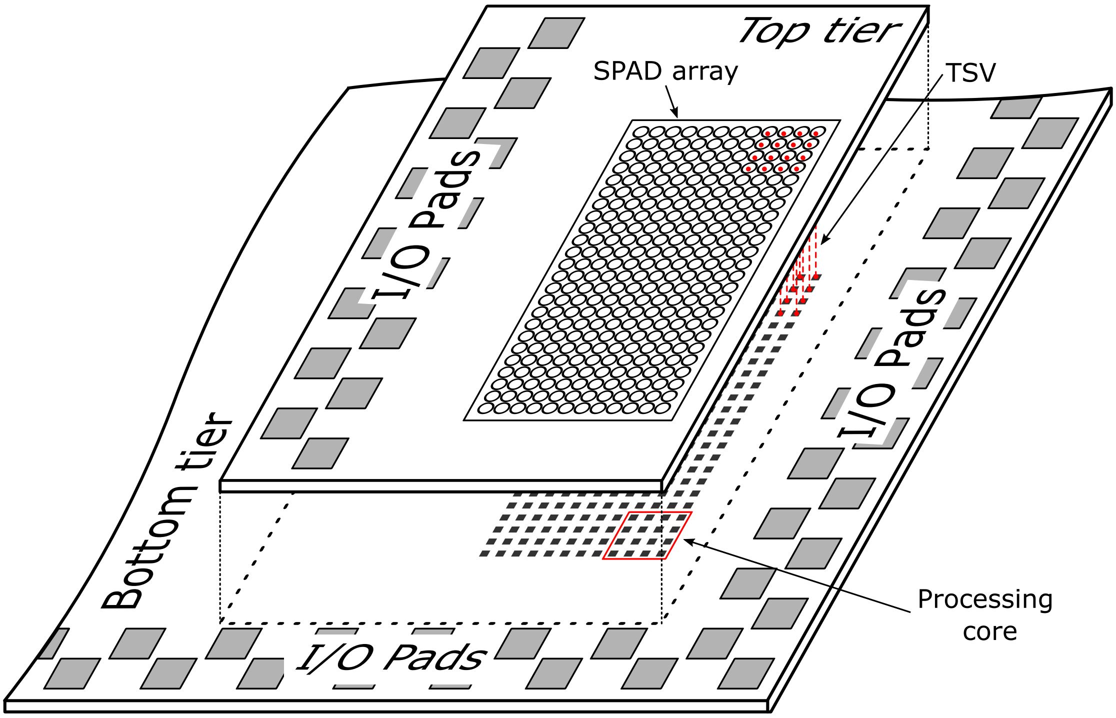


Figure 4.8: *Ultra*Phase 3D structure. The top tier is smaller than the bottom to allow access to I/O pads. TSVs provide the connection between landing sites on the bottom tier and the SPADs on the top.

4.3.1 Top tier

The top tier was fabricated in 180 nm CMOS. Due to the lack of any pixel front end circuitry on the bottom tier, the top tier contains both the SPAD and the transistors required to operate it. In addition, a voltage level translation was required to adapt from the 1.8 V pixel to the 0.8 V logic.

Figure 4.9 shows the pixel schematic. Similarly to *kilo*Phase and *Mega*Phase, cascode transistor T_1 is used to extend the bias voltage range of the pixel while transistor T_2 implements the clock-driven active recharge controlled by the RST signal. When an avalanche takes place, the voltage at point A will rise and depending on the state of gate transistor T_3 can act on point B and transistor T_5. If node B is high, T_5 will drive the gate of oversized transistor T_8 that discharges the large parasitic capacitance of the

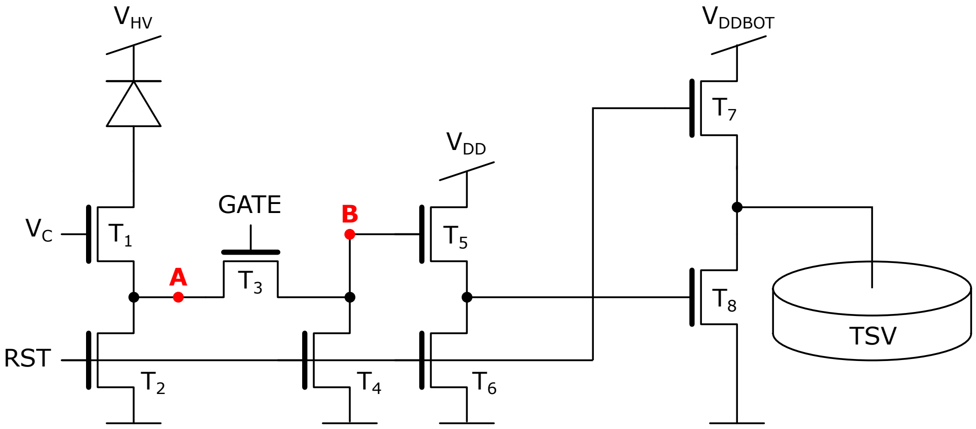


Figure 4.9: *Ultra*Phase top tier pixel schematic. All the transistors are thick oxide NMOS which reduces the size of the layout by eliminating the large spacing requirements between wells. Transistor T_1 is used to extend the SPAD bias voltage range. T_2 implements a clock-driven recharge scheme and T_3 is used to gate the pixel. Transistors T_7 and T_8 are used to charge and discharge the large parasitic capacitance of the TSV that is used as a 1 bit pixel memory. T_4 and T_6 are reset transistors for intermediate nodes. V_{DDBOT} is used to implement voltage translation, as the bottom tier operates at a lower supply voltage.

TSV. The RST signal also drives transistors T_4, T_6 and T_7 used to reset node B and the gate of T_8 and to recharge the TSV capacitance. Voltage level translation is achieved by setting supply voltage V_{DDBOT} to 0.8 V plus the threshold voltage of the thick oxide transistor.

The same SPAD model as the one used in Chapter 2.4 was used here as well for all of the pixel simulations. The parasitic capacitance of the TSV was estimated based on the work presented in [147] that uses the same technology and process. The large parasitic capacitance of the TSV was used as a memory cell for the pixel and was validated in the Fast-Fast library simulation corner to assure that data retention is long enough to not compromise sensor operation. Transistors T_7 and T_8 were sized so that charging and discharging the TSV capacitance takes less than 400 ps in all simulation corners (Typical, Fast-Fast and Slow-Slow). All of the simulations were performed in SPICE using post layout parasitic values.

Figure 4.10 shows the pixel layout, designed for a 28.5 µm pitch at a 29.5 % fill factor. The octagonal shape at the bottom left is the metal contact for the 8 µm diameter TSV. All the transistors are located in the bottom and left side rectangular sections. Due to the small area and spacing requirements, the transistors are all thick oxide NMOS. The SPAD structure is identical to the one used for *kilo*Phase but with a slightly larger active area diameter of 15.45 µm. Validation of the layout was done manually because

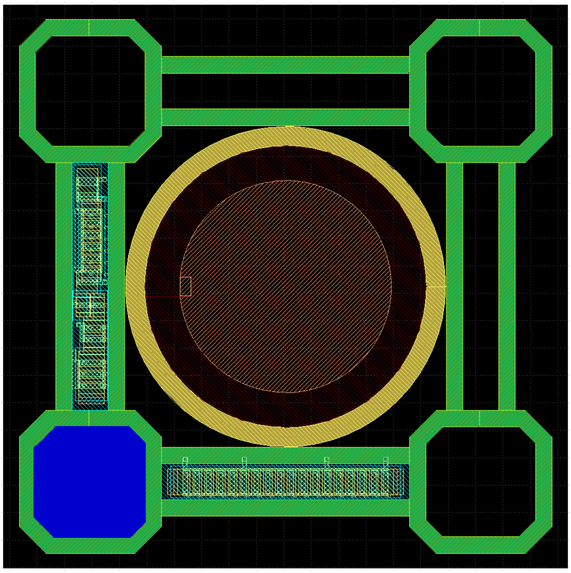


Figure 4.10: *Ultra*Phase top tier pixel layout, showing the SPAD in the middle, with the TSV in the bottom left. All the transistors are contained in the bottom and left sections. The pixel is designed to be abutted to form the desired array size at a 28.5 µm pitch.

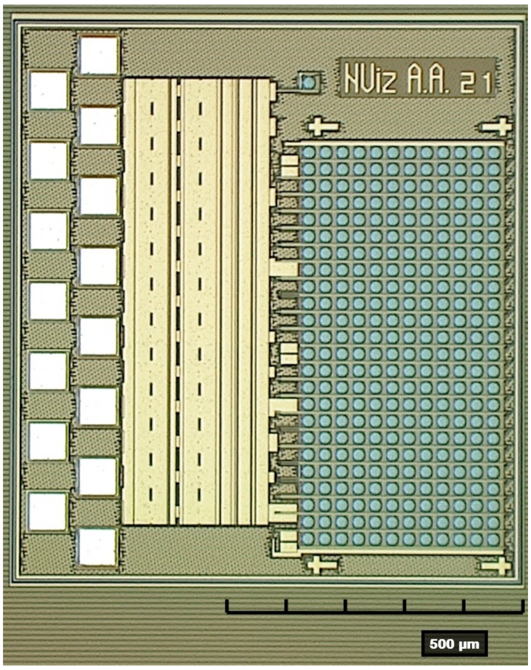


Figure 4.11: *Ultra*Phase top tier micrograph. The SPAD array is clearly visible on the right, together with the 4 cross-shaped alignment markers used for 3D stacking. The left side contains the IO pads for the supply and bias voltages as well as connections dedicated to a standalone test pixel. The die measures 864 × 954 µm^2.

the LVS tool could not recognize the SPAD structure and it had to be replaced by a black box.

A micrograph of the fabricated top tier is shown in Figure 4.11. The 12 × 24 SPAD array is visible on the right, together with the 4 cross-shaped alignment markers used for the 3D stacking procedure. The supply and bias voltages for the top tier, along with the gate and reset signals are provided through 15 I/O pads. A standalone test pixel with its own dedicated pads was placed on the top tier for pixel characterization. The final size of the top tier die is 864 × 954 μm^2. Dummy metal generation was performed on the entire design with the areas above the SPADs being blocked by special-purpose layers to prevent any obstructions.

4.3.2 Bottom tier

The bottom tier was fully synthesized with Cadence Genus and then placed and routed with Cadence Innovus. A hierarchical approach was used in order to simplify the process and reduce the tool resource requirements. The processing core was synthesized and routed as a stand-alone structure which was then imported as a hard macro into the *Ultra*Phase top level which also contained additional glue logic such as an AXI interface and registers. The complete design was then delivered as a macro and was integrated together with other contributions from EPFL and Bar-Ilan University into a SoC built around a Pulpino RISC-V core [148]. A system-wide AXI bus provides connections with all the contributions for reading and writing data, but apart from this, *Ultra*Phase is completely independent and does not require any external circuitry.

The processing core layout is shown in Figure 4.12. Each core is 107 × 107 μm^2 and has a 4 × 4 28.5 μm pitch grid of metal pads drawn on top. The pads are designed to act as landing sites for the TSVs used in the 3D integration process and are electrically connected to the pixel inputs of the core and ESD protection diodes placed using the Cadence tool.

Figure 4.13 shows a micrograph of the entire SoC die with the inset showing a corner of the *Ultra*Phase bottom tier, where the TSV landing sites can be seen next to the normal size bonding pads. The placement of the cores followed a pitch that would ensure uniformity of the resulting 12 × 24 grid of 16 × 16 μm^2 rectangular aluminium contacts for the TSVs. The cross shaped element is an alignment marker used during the 3D stacking procedure.

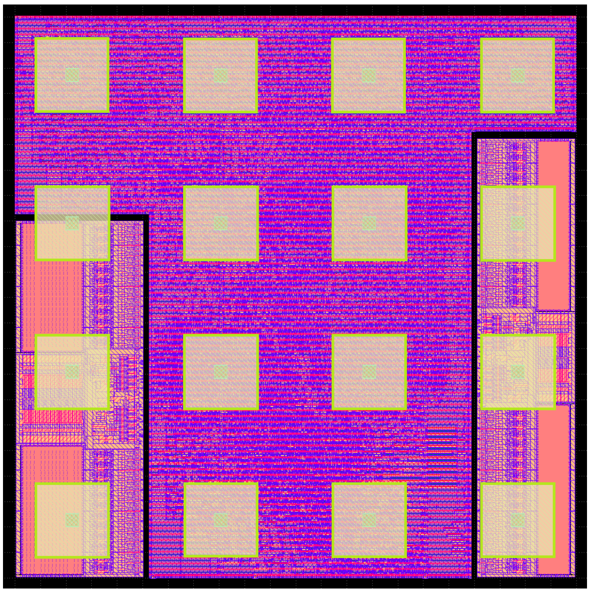


Figure 4.12: *Ultra*Phase processing core layout. The two memories are visible at the left and right edges of the layout, occupying approximately 30% of the available area. The 4×4 grid of metal pads for 3D integration is shown in green. The core is $107 \times 107\ \mu m^2$ in size.

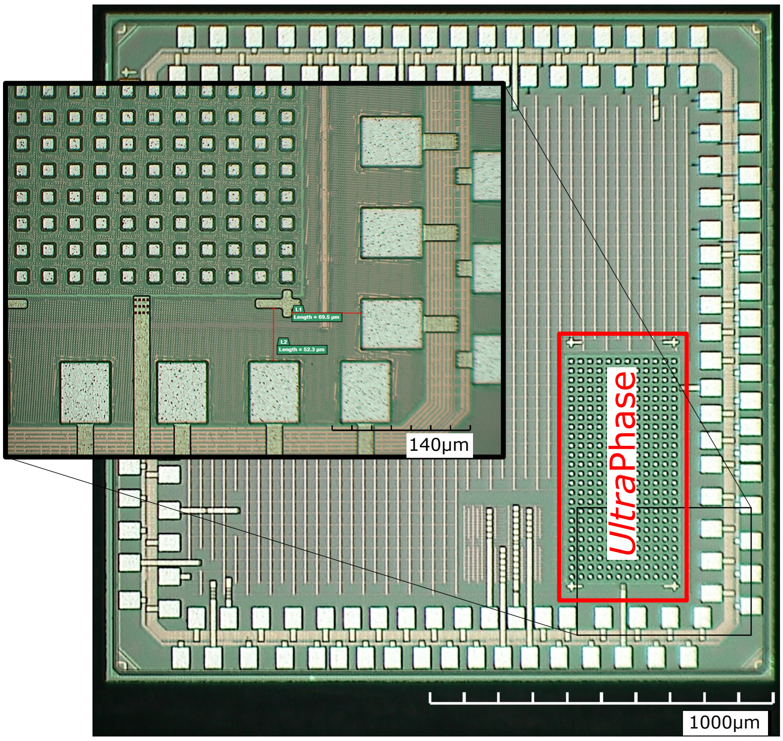


Figure 4.13: Bottom tier SoC micrograph with *Ultra*Phase visible in the bottom right. The inset is a close-up of the chip corner containing TSV landing sites and standard I/O pads. The cross-shaped element is an alignment marker.

Memory blocks

All the digital processing elements are located in the middle of the core with the two memory blocks occupying approximatively 30% of the available area placed along the left and right edges of the core. The memories were generated as macros using a tool provided by the foundry based on the *Ultra*Phase design specifications. Their placement in the core reduces the risk of routing congestion at the memory interface caused by the large number of connections and the routing restrictions above the memory cells. Placement of the standard cells was restricted to a density of 50% in the area surrounding the memory ports as a method of reducing congestion even further.

Clock distribution and timing

Routing of the processing core was limited to the first 5 metals. The system clock was routed using metals M3 and M4 with double spacing constraints to limit parasitic coupling to any neighboring signals. Timing closure was obtained at 500 MHz in all 20 library corners shown in Table 4.4. The complete *Ultra*Phase array of processing cores was routed using the first 7 metals in the stack and targeted a maximum operating frequency of 400 MHz that was achieved in all corners.

Power distribution

A dense grid of M5 strips was used to distribute power across the entire processing core and to shield it from any possible interference from signals routed above it. The M5 strips periodically connect to the M1 metal in the standard cells to assure a low impedance connection. The *Ultra*Phase array was then built by tiling the processing core macros and routing another dense power distribution grid using metals M6 and M7 that directly connected to the M5 stripes from each core. Finally, at the SoC level, the *Ultra*Phase power grid was connected to wide metal rings routed around the macro that provided the necessary low impedance path to the power pads.

As this is an advanced technology node, leakage currents in the decoupling capacitors are non-negligible and have to be carefully mitigated. In this case, decoupling cells were placed only in key points of the design: around the memory macros and in close proximity to the clock distribution network. The majority of the remaining space in the design was left empty.

Table 4.4: All the library corners used for the *Ultra*Phase timing analysis.

Corner	Transistor model	Voltage	Temperature	Interconnect	Coupling
1	Slow-Slow	0.72 V	125 ^oC	C-worst	worst
2	Slow-Slow	0.72 V	125 ^oC	RC-worst	worst
3	Slow-Slow	0.72 V	0 ^oC	C-worst	worst
4	Slow-Slow	0.72 V	0 ^oC	RC-worst	worst
5	Slow-Slow	0.72 V	125 ^oC	C-worst	worst
6	Slow-Slow	0.72 V	125 ^oC	RC-worst	worst
7	Slow-Slow	0.72 V	0 ^oC	C-worst	worst
8	Slow-Slow	0.72 V	0 ^oC	RC-worst	worst
9	Fast-Fast	0.88 V	125 ^oC	C-worst	worst
10	Fast-Fast	0.88 V	125 ^oC	C-best	best
11	Fast-Fast	0.88 V	125 ^oC	RC-worst	worst
12	Fast-Fast	0.88 V	125 ^oC	RC-best	best
13	Fast-Fast	0.88 V	0 ^oC	C-worst	worst
14	Fast-Fast	0.88 V	0 ^oC	C-best	best
15	Fast-Fast	0.88 V	0 ^oC	RC-worst	worst
16	Fast-Fast	0.88 V	0 ^oC	RC-best	best
17	Typical-Typical	0.8 V	85 ^oC	C-worst	worst
18	Typical-Typical	0.8 V	85 ^oC	RC-worst	worst
19	Typical-Typical	0.8 V	85 ^oC	C-best	best
20	Typical-Typical	0.8 V	85 ^oC	RC-best	best

Dummy generation

Dummy metal generation required special attention because the parasitic elements in such an advanced technology node could have a drastic impact on the timing performance of the design. Once the processing core layout was complete and the static timing analysis reported no violations across all 20 corners, the signals with a slack below a certain threshold were marked as *victims* and the dummy metal spacing rules around them were doubled. Dummy generation was performed up to metal M5, after which the timing analysis was repeated.

At the array level, the processing core macros each containing their own dummy patterns were covered with blocking layers on the first 5 metals so that the next dummy generation step would not change the timing performance. Dummy generation was performed up to metal M7 with no risk of affecting the processing cores as they were shielded from M6 and M7 patterns by their dense power distribution network. The *Ultra*Phase design was then delivered as a macro for integration into the SoC and the process was repeated for the remaining metal stack.

Validation

The design was validated by running digital simulations in Cadence SimVision. The Verilog netlist of the entire SoC was exported from Cadence Innovus together with the .sdf file containing the net propagation delays for corners 2, 16 and 18. SPAD inputs were provided by a Verilog testbench. Multiple test programs were run on *Ultra*Phase to confirm the validity of the architecture. The same simulations were repeated at various stages of the design, starting with behavioral simulations and ending with gate level simulations with parasitics.

4.4 Programming

Programming of each core is performed by uploading the program into the ROM through the AXI bus. In order to avoid any unexpected behaviour, the targeted core should be kept in the reset state during this procedure. However, there are two exceptions when a core can be reprogrammed at runtime: if there is certainty that the execution of a certain part of the program will not take place until reprogramming has finished or if the core is kept frozen waiting for an external stimulus using the special-purpose WAIT instruction.

4.4.1 Instructions

In total, there are 44 instruction types that can be classified into 5 categories: logic, arithmetic, manipulation, flow and special. The majority of instructions have multiple variants depending on the source of the operands and the destination of the result. Table 4.5 details all the possibilities. When counting all the possible variants, there are 292 instructions. The shift and compare instructions were inspired from the Xilinx PicoBlaze instruction set [149].

Logic instructions

Logic instructions can have one or two operands sourced from either a register or a RAM location. The result can be written to any register or RAM location, including one that acted as a source. This type of operation does not support explicit operands because it would require a 32 bit payload, larger than the instruction width. After execution, the ALU carry flag will be cleared regardless of the result, these instructions replacing a dedicated clear flag command. The ALU zero flag functions as normal. The four bitwise logical operations supported by the architecture are: NOT, AND, OR

Table 4.5: The possible operand source and result destination for *Ultra*Phase instructions.

Operand A	Operand B	Result
Register	Register	Register
Register	Register	Absolute RAM Address
Register	Register	Relative RAM Address
Register	Absolute RAM Address	Register
Register	Absolute RAM Address	Absolute RAM Address
Register	Absolute RAM Address	Relative RAM Address
Register	Relative RAM Address	Register
Register	Relative RAM Address	Absolute RAM Address
Register	Relative RAM Address	Relative RAM Address
Register	Explicit value	Register
Register	-	Neighbor
Neighbor	-	Register

and XOR.

Arithmetic instructions

There are a total of 8 arithmetic instructions that can be performed by the ALU: sign inversion, addition, subtraction, multiplication, MAC, MAX, MIN and value comparison. There is no dedicated operation for division or nonlinear functions. Similarly to the logical instructions, they can act on data from the general purpose registers or the RAM, but can also use three operands (MAC instruction) or explicit values (ADD and SUB instructions).

All the operations act on both the carry and zero flags, but the compare (CMP) instruction is a special case because the flags do not maintain their normal behaviour and there is no instruction output. Table 4.6 shows the behaviour of the two flags when the CMP instruction is used on operands A and B.

Table 4.6: The effects of the CMP instruction on the ALU flags.

Condition	Zero flag	Carry flag
A <B	0	1
A = B	1	0
A >B	0	0

Manipulation instructions

Manipulation instructions act on a single operand and are used to apply rotations and shifts or select specific bytes from the 32 bit word. The category also contains the RAM STORE and FETCH instructions that transfer data from a general purpose register to the RAM or, in reverse, the latter option supporting multiple destinations at the same time. In addition, a LOAD instruction is provided to write an explicit value to any of the general purpose registers. However, because the width of the instruction code is smaller than that of the register, the instruction only writes two bytes at a time and must be called twice for a complete register initialization.

Flow instructions

Flow instructions influence the execution of the program by changing the PC register. The JUMP instruction can be used to jump to any address in the instruction memory either unconditionally, or depending on the status of the available flags. The CALL and RET instructions are used to execute subroutines. The former acts exactly like the JUMP instruction and its variants, but will push the PC value to the stack so that when RET is called, program execution can resume from the same point.

Special instructions

The highly customized architecture requires a special set of instructions that are not normally encountered with other CPUs. Communication with the neighbors and the external circuits is done through the SAVEN, GETN, PUTN and TELL instructions. The first two are used to sample the neighbor data bus and transfer the value to the general purpose registers. Both instructions support multiple sources and destinations at the same time. The PUTN instruction will latch the value from a general purpose register onto one or multiple neighbor data buses. Finally, TELL is used to strobe or set synchronisation signals for the neighboring control modules or the external I/O pads.

Data from the timing module or the front end can be read using dedicated GETC and GETP instructions that support simultaneous byte selection and multiple destinations.

All the combinational logic paths have their own configuration instructions, starting with the front end multiplexers (SETFM, SETTM) and the LUT functionality (SETLUT) and ending with the fast path OR tree (SETOR) and timing module (SETTIME).

Finally, a special WAIT instruction is provided to facilitate the simultaneous synchronisation of the cores with an asynchronous external trigger condition. When running this instruction, the Control module is frozen in the execute stage until the specified condition is met, after which operation resumes immediately, at the next clock cycle. The condition is verified by monitoring the neighbor and external synchronization signals and is only met when a pulse has been detected from all of the requested sources, regardless of order.

4.4.2 Assembler

An assembler software was written in Python 3.6. It takes as input a .txt file containing the program code written in the custom *Ultra*Phase assembly language and outputs a .csv file with the corresponding machine code that serves as the memory initialization file. As a safety feature, any unused memory locations contain a jump instruction to themselves, causing the core to remain stuck in an infinite loop, thus eliminating the possibility of corrupt program execution.

Syntax of the assembly language was kept as simple as possible by using a standard structure for each line:

$label: \quad INSTRUCTION \quad operand1, \quad operand2, \quad operand3 \quad --comment$

$Label$ is the address in the ROM where the instruction will be written to, and must be manually determined by the programmer, the only restriction being that the first instruction in the program must have the label equal to 0. $INSTRUCTION$ is a keyword specific to each operation described in section 4.4.1 and can be used by itself or followed by one, two or three operands. An optional comment can be written after the special character sequence $--$.

Multiple special $\#define$ directives can be placed at the beginning of the assembly file to designate suggestive names to constants and registers used in the program. The syntax to replace every *alias* in the program with *value* is:

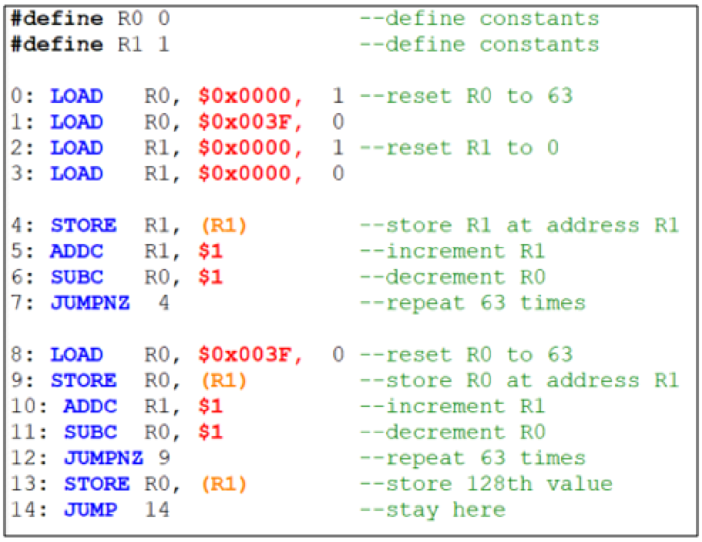
$$\#define \quad alias \quad value \quad --comment$$

There are three special characters recognised by the assembler that change the interpretation of a specific instruction:

- The *$* character written in front of an operand signifies that it represents an explicit numerical value instead of a register address.

- The @ character indicates that the following operand is an absolute address in the RAM.
- If an operand is enclosed between *( )* it will be used as a pointer i.e. the value stored in the register is treated as a RAM address.

In order to improve legibility of the assembly code, a user-defined language XML file was written for Notepad++ users. Importing the file into the text editor will highlight key assembly words such as the instruction names and operands as shown in Figure 4.14.



```
#define R0 0              --define constants
#define R1 1              --define constants

0: LOAD   R0, $0x0000,  1 --reset R0 to 63
1: LOAD   R0, $0x003F,  0
2: LOAD   R1, $0x0000,  1 --reset R1 to 0
3: LOAD   R1, $0x0000,  0

4: STORE  R1, (R1)        --store R1 at address R1
5: ADDC   R1, $1          --increment R1
6: SUBC   R0, $1          --decrement R0
7: JUMPNZ  4              --repeat 63 times

8: LOAD   R0, $0x003F,  0 --reset R0 to 63
9: STORE  R0, (R1)        --store R0 at address R1
10: ADDC  R1, $1          --increment R1
11: SUBC  R0, $1          --decrement R0
12: JUMPNZ 9              --repeat 63 times
13: STORE R0, (R1)        --store 128th value
14: JUMP  14              --stay here
```

Figure 4.14: *Ultra*Phase assembly language as seen in Notepad++ when the custom language XML file is used. Instructions are highlighted in blue, comments in green, define statements in black and numerical values in red.

4.4.3 Compiler

While the assembler provides an efficient way to optimize programs due to the ability to interact directly with the hardware, it is not a preferable solution for quickly implementing code that is easy to read and to debug. For this task, a higher level programming language is better suited.

A C compiler was written in Python 3.6, based on the open source ShivyC Linux x86-84 architecture compiler [150]. All the work was done by Yining Zhu from Queen Mary University of London, as part of his bachelor internship, under the author's supervision.

The compiler only supports int8, int16 and int32 data types due to hardware limitations. Division is currently not handled and should be avoided. The limited hardware stack size was enforced by dedicating one of the general purpose registers to act as a counter for recursive subroutine calls. In case the limit is exceeded, the PC will jump to the end of the program and stop.

The custom instructions are accessed in C by using setter and getter functions, such as *set_time_mux(val)* to call SETTM or *get_pixel(&val, source)* to call GETP, or by using the *__asm* directive followed by the assembler instruction.

Future improvements of the compiler should focus on a more efficient way of developing parallel code that can run on the 3 × 6 array of processors. The task is currently being performed by writing code for each individual core and manually taking care of all the inter-core communication, which makes it prone to errors.

4.5 Characterization

4.5.1 Power consumption

Processor power consumption was measured for multiple test cases by running a continuous code loop. Leakage power was measured by gating the global clock signal and then the value was subtracted from the other measurements. Standby power was measured during execution of an empty *while* loop. Two types of MAC instructions, one operating only with data from the registers and the other with data from the RAM were measured in order to characterize the ALU power consumption. Finally, the timing module was configured to count the number of oscillations of the local oscillator and the periods of the system clock. The results are presented in Figure 4.15.

As expected, the largest power consumption is observed when executing the MAC instruction that requires reading from and writing to the RAM. The other operations display similar power consumption, apart from broadcasting the data to the direct neighbors, which is two orders of magnitude more efficient. This is due to the significant design effort made to keep the wide interconnect buses between the cores as short as possible, reducing the load on the output buffers.

Each core can operate at a maximum performance of 140 MOPS resulting in a worst case power consumption of 94.6 GOPS/W/core.

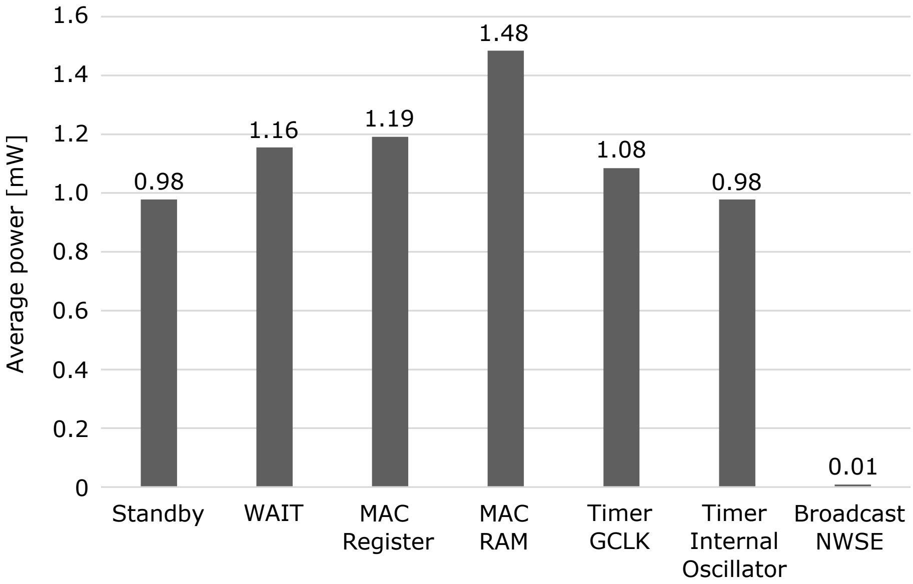


Figure 4.15: Average *Ultra*Phase core power consumption for multiple test cases.

4.5.2 Timing performance

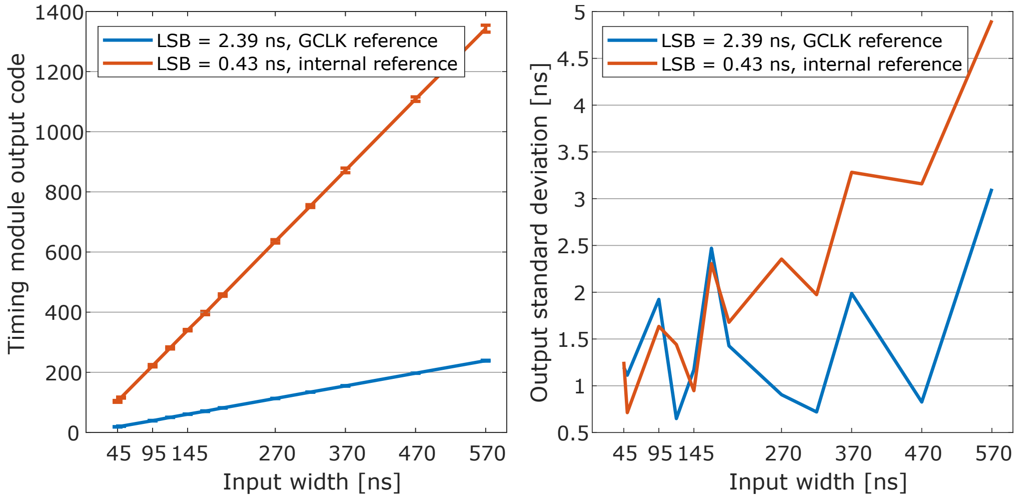
The timing module was characterized by measuring the number of completed oscillations of the internal oscillator during a well-defined time interval referenced to the 420 MHz global clock. The measurement was performed at the nominal supply voltage of 0.8 V and at the two extreme values of 0.88 V and 0.72 V. The results are summarized in Table 4.7.

Table 4.7: Timing module resolution obtained with the internal reference oscillator.

Supply voltage [V]	**LSB [ps]**
0.72	649
0.8	433
0.88	340

Single-shot measurements were performed with the timing module configured to measure the delay between two input pulses generated by an FPGA. Two cases were analyzed, with the local oscillator and the global clock serving as references. The significant difference between the two is the fact that the latter is generated by a PLL and as a result has much lower phase noise. The delay between the input signals was changed from 45 ns to 570 ns. 1270 measurements were performed at each step.

Figure 4.16 shows the resulting transfer functions and the standard deviations of the outputs for each input width. As expected, the unregulated oscillation of the internal reference accumulates phase error at large input pulse widths, but performs similarly to the PLL-stabilized global clock when operating in the short range.



(a) (b)

Figure 4.16: *Ultra*Phase timing module single shot measurements: (a) Transfer functions of the timing module for the two reference clocks. (b) Standard deviation of the timing module outputs for each input width. Phase noise accumulation causes a performance degradation when operating with the fast internal oscillator and large input pulse widths. All of the measurements were performed at room temperature.

4.6 Applications

4.6.1 ToF histogram compression

The following application was developed in collaboration with WISIONLab, University of Wisconsin-Madison, based on [151]. I was responsible for porting the algorithm to the *Ultra*Phase architecture and as a result, the chapter will only present information relevant to my work.

Direct time of flight measurements consist of illuminating scenes with pulses of light and collecting arrival timestamps from photons reflected off of the items in the scene. The timestamps are then used to build histograms which are further processed to extract information such as depth.

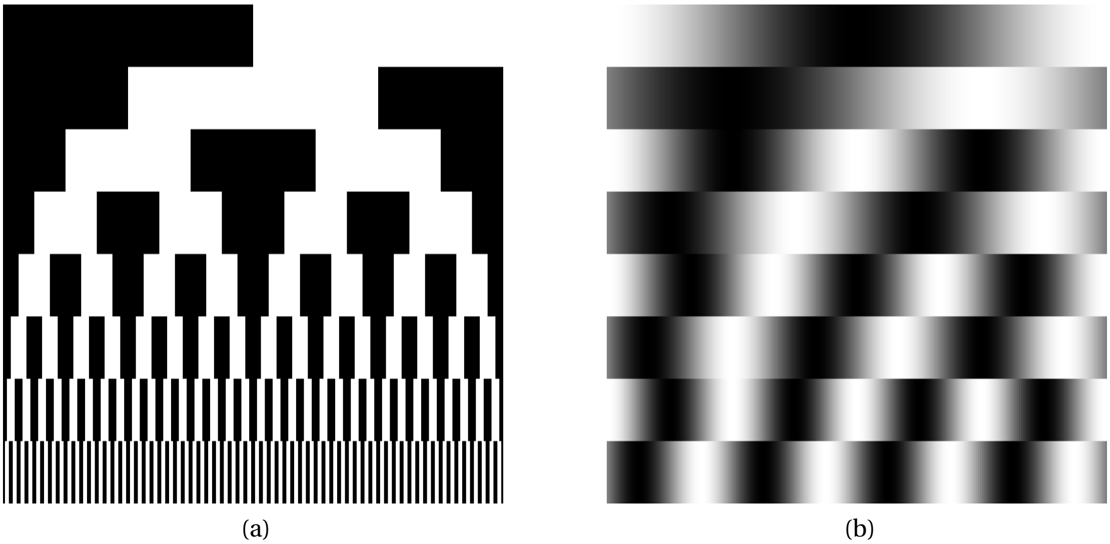
As the resolution of timing circuits increases, the number of bins in the histograms grows larger and becomes difficult to handle. Advanced sensor architectures have already been developed to mitigate the problem by using peak detection or gating algorithms that only preserve histogram sections of interest [38], [64], [130], [152],

[153].

A different approach is being investigated, where the histogram data is encoded into a smaller number of variables that can then be used to extract the relevant information from the original histogram [151], [154]–[157]. *Ultra*Phase is a good candidate platform for testing and developing the encoding algorithms because of its flexibility and parallel computational power.

For demonstration purposes, the timing module of each core was configured to function as a time to digital converter (TDC) and measure the number of clock cycles between the arrival of a photon at any of the associated 16 pixel inputs and a known reference signal. The exposure window was chosen such that the TDC range was limited to 8 bit.

The encoding algorithm works by converting 8 bit timestamp t into 8 variables b_i which are then accumulated separately. After a large number of cycles, the 8 accumulated results are read from the chip and processed to recover the scene information [151], [154], [155]. The encoding function is represented in this case by an 8 × 256 matrix of 16 bit numbers. Timestamp t is used as an index to extract a column from the matrix that contains the 8 encoding elements that have to be accumulated. Selecting the appropriate encoding matrix is a subject of its own, but two common examples are Gray or Fourier codes, as shown in Figure 4.17 [151].


(a)

(b)

Figure 4.17: Gray (a) and Fourier (b) 8 × 256 16 bit encoding matrices. An 8 bit timestamp is used as a column index to select a group of 8 values that are used for further processing.

If the matrices shown in Figure 4.17 were to be used in hardware, they would each occupy 32.768 kbit of RAM, far exceeding the available resources from an *Ultra*Phase core. The problem can be mitigated by distributing one matrix across multiple cores and using inter-core communication to transmit the timestamps to the core that contains the relevant section for processing. However, a better solution is to take advantage of the processing power in each core and compute the necessary encoding coefficients for the current timestamp at every cycle, eliminating the need for storing the encoding matrix all together.

The 8 bit Gray code representation t_{gray} of timestamp t can be easily computed using:

$$t_{gray} = t\, XOR\, (t >> 1), \tag{4.3}$$

where >> represents the shift right operation. From this value, the 8 encoding variables b_i are given by:

$$b_i = \begin{cases} 1, & when\ t_{gray}[i] = 1 \\ -1, & else. \end{cases}$$

The Gray encoding-based implementation in *Ultra*Phase occupies 12.5% of the available instruction ROM and 1.56% of the RAM. It requires 507.1 ns to run a full cycle, 21% of which is the exposure time, assuming that the fast local oscillator in the timing module is used. As each core operates independently, the processing time remains constant regardless of the size of the array, the only frame rate limitation being the core readout time which is approximately 571 ns/core.

The Fourier encoding is more demanding as it requires the use of trigonometric functions and normalization of timestamp t to the $[0, 2\pi]$ interval:

$$\begin{aligned} b_{2k} &= cos\left(\frac{2(k+1)\pi t}{255}\right) \\ b_{2k+1} &= -sin\left(\frac{2(k+1)\pi t}{255}\right), \end{aligned} \tag{4.4}$$

where $k \in [0, 3]$. The normalization and trigonometric functions can be implemented as a LUT. However, due to the timestamp range, the total memory required to store even an 8 bit LUT would be 4.096 kbit, or the entire core RAM. The solution in this case is to limit the LUT to cosine values for the input range $[0, \frac{\pi}{2}]$ and use trigonometric identities to compute the remaining ones:

$$cos(t) = \begin{cases} cos(t), & when\ t \in [0, \frac{\pi}{2}] \\ -cos(\pi - t), & when\ t \in (\frac{\pi}{2}, \pi] \\ -cos(t - \pi), & when\ t \in (\pi, \frac{3\pi}{2}] \\ cos(2\pi - t), & when\ t \in (\frac{3\pi}{2}, 2\pi]. \end{cases} \tag{4.5}$$

The sine value can also be determined from the same LUT based on:

$$-sin(t) = cos(t + \frac{\pi}{2}). \tag{4.6}$$

The Fourier encoding-based implementation in *Ultra*Phase occupies 29.3% of the available ROM. The LUT requires 50% of the RAM but can be packed more efficiently if needed. The total runtime for a full cycle is 607.1 ns, out of which 17.6% is the exposure time, under the same assumptions as before.

Table 4.8: Gray and Fourier code histogram encoding performance

	Gray code		**Fourier code**	
Number of coefficients	8	16	8	16
Exposure time	107.1 ns	27.4 µs	107.1 ns	107.1 ns
Processing time	400 ns	757.1 ns	500 ns	785.7 ns
Frame rate[a]	19.5 kfps	354.7 fps	16.3 kfps	11 kfps
Core output data rate[a]	5 Mb/s	181.6 kb/s	4.2 Mb/s	5.6 Mb/s
Data rate reduction[a,b]	÷3.1	÷3.1	÷3.1	÷1.6
Data compression factor[a,c]	8	1024	8	4

[a] Assuming 100 cycles per frame.
[b] Compared with outputting the raw TDC timestamps at the same frame rate on a parallel bus.
[c] Compared with storing all the bins of the 8 bit histogram.

In both cases, extension to 16 coefficients is trivial and does not change the length of the programs, just the execution times. Table 4.8 summarizes the results for both Gray and Fourier code encoding, for 8 and 16 coefficients. Extending the Gray code implementation to 16 coefficients requires the use of a 16 bit TDC and a larger exposure time. This is not needed for the Fourier implementation where more coefficients can be calculated using the same 8 bit timestamp. As a result, the benefits in terms of compression and data rate reduction are not as large for the 16 bit Fourier implementation example.

4.6.2 LSTM LiDAR

The following application was developed in collaboration with Tommaso Milanese from EPFL and I was responsible for porting the algorithm to the *Ultra*Phase architecture. As a result, the chapter will only present information relevant to my work.

A long short-term memory (LSTM) is a special type of artificial neural network that contains feedback connections which allow the processing of data sequences such as audio or video signals [158]. Recently, an idea of extending the use of LSTM to LiDAR applications has emerged, where the data stream generated by the time of flight (ToF) image sensor is processed by such a network to determine the depth map of a scene, eliminating the need for histogramming. The unique properties of *Ultra*Phase make it a very good candidate for this application for a couple of reasons: firstly, the ability of the processing cores to share information between them allows for a high degree of parallelisation and secondly, the reconfigurable front end can implement preprocessing techniques such as coincidence detection with no speed or processing penalty.

For demonstration purposes, the following scenario was chosen: the sensor acts as a single point ToF detector used in an X-Y scanning setup and implements an LSTM cell of size 20. All the computations are performed using signed fixed point representations with 3 fractional bits. The front end is configured to trigger the timing module with the first detected input pulse within an exposure window.

Figure 4.18 shows the diagram of a LSTM cell, the state of which is described at

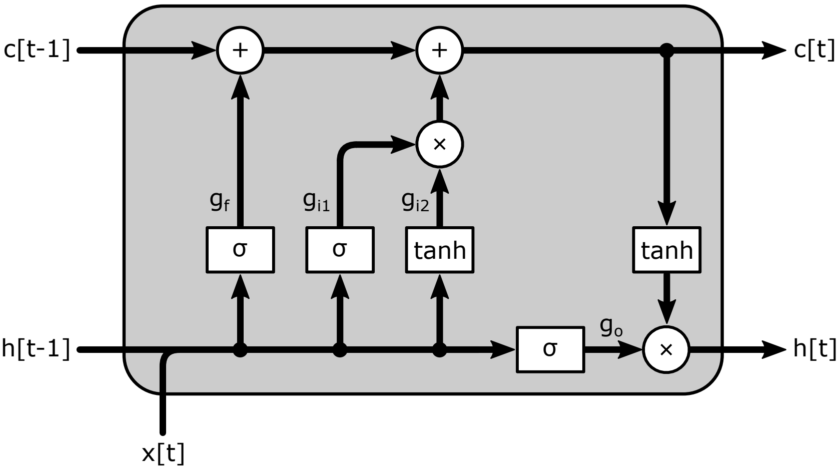


Figure 4.18: LSTM cell structure. At every time moment t, two system variables c and h are computed based on their value in the previous cycle at time $t-1$ and new input data x. Two nonlinear functions, the sigmoid and hyperbolic tangent, are used as activations.

time step t by the following equations:

$$\begin{aligned} g_f &= \sigma\left(W_f \times x[t]||h[t-1] + b_f\right) \\ g_{i1} &= \sigma\left(W_{i1} \times x[t]||h[t-1] + b_{i1}\right) \\ g_{i2} &= \tanh\left(W_{i2} \times x[t]||h[t-1] + b_{i2}\right) \\ g_o &= \sigma\left(W_o \times x[t]||h[t-1] + b_o\right) \\ c[t] &= g_f \cdot c[t-1] + g_{i1} \cdot g_{i2} \\ h[t] &= g_o \cdot \tanh(c[t]), \end{aligned} \tag{4.7}$$

where g_f, g_{i1}, g_{i2} and g_o are 20×1 arrays of values for the *forget*, *input* and *output* gates, h and c are 20×1 vectors representing the *hidden* and *cell* states that are saved from one LSTM iteration to the next, x is the input value given by the TDC, W_f, W_{i1}, W_{i2} and W_o are 20×21 weight matrices and b_f, b_{i1}, b_{i2} and b_o are 20×1 bias arrays. All the W and b values are constant and determined before runtime during the training of the LSTM. σ and tanh are the sigmoid and hyperbolic tangent functions while $||$, $\times$ and $\cdot$ represent the concatenation, matrix multiplication and Hadamard multiplication.

Figure 4.19 presents the scheduled graph for equations (4.7). Step S_0 is trivial, as the concatenation operation can be replaced by a memory write. Steps S_1 and S_2 are the most resource intensive, requiring a total of 420 fixed point MACs. Steps S_4, S_5 and S_7 only require 40 fixed point multiplications and 20 fixed point additions/multiplications respectively. The nonlinear tanh and σ functions can be implemented as LUTs, i.e. simple memory read operations. In order to increase the execution speed, steps

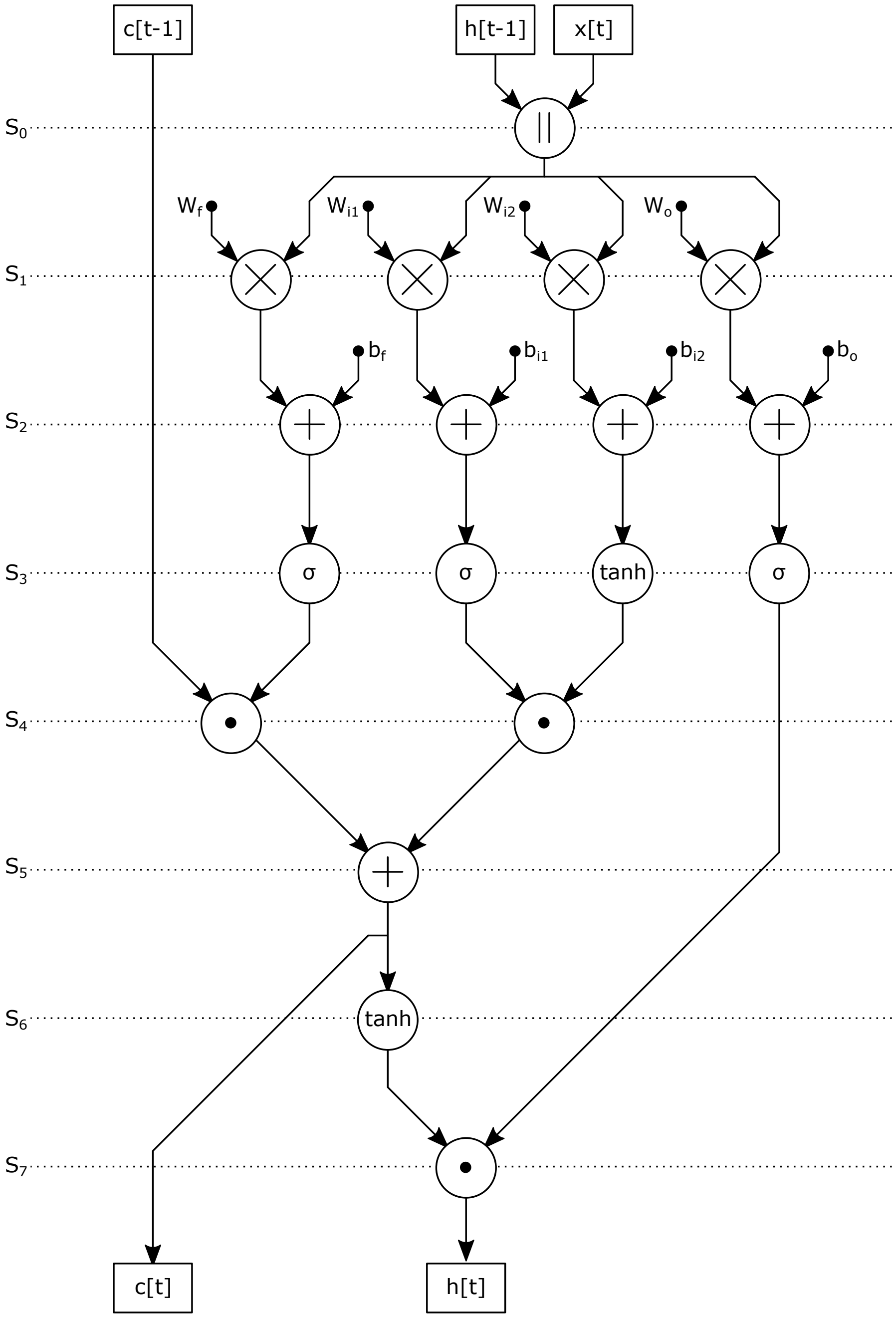


Figure 4.19: LSTM scheduling graph

S_1 and S_2 are distributed across 4 separate slave cores, while the remaining steps are assigned to a single master core.

The total available RAM in each core is 512 bytes and as a result, all weight and bias coefficients have to be stored as 8 bit signed fixed point numbers with 3 fractional bits, as 4 coefficients per RAM word. The g_f, g_{i1}, g_{i2}, g_o, h and c values have a 16 bit signed fixed point representation with 3 fractional bits and are stored in pairs at each RAM location. The LUTs used for the nonlinear activation functions have the same data format as the previous variables. Table 4.9 summarizes the total RAM usage for the master and slave core.

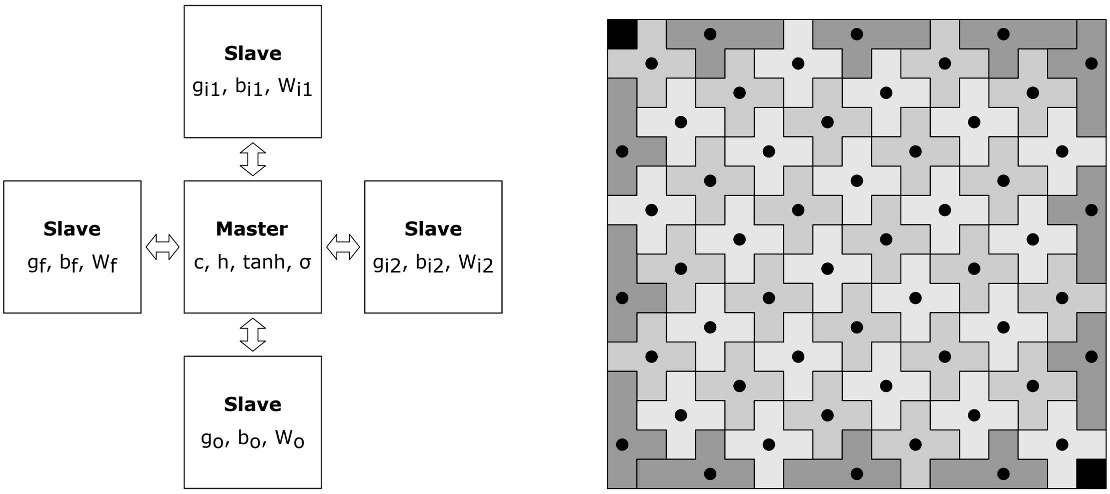
Table 4.9: RAM usage per core for the LSTM implementation.

Master core RAM			**Slave core RAM**		
Variable	Size		Variable	Size	
c	40 byte	7.81 %	h	40 byte	7.81 %
h	40 byte	7.81 %	x	2 byte	0.39 %
g_f	40 byte	7.81 %	b	20 byte	3.91 %
g_{i1}	40 byte	7.81 %	W	420 byte	82.03 %
g_{i2}	40 byte	7.81 %			
g_o	40 byte	7.81 %			
tanh & σ LUTs	256 byte	50 %			
Total	**496 byte**	**96.88 %**	**Total**	**482 byte**	**94.14 %**

Figure 4.20a shows the arrangement of the 5 computational units used for the LSTM implementation. The master core is surrounded by the slave cores in the four cardinal directions in order to allow the fastest data transfer possible. Once the master core finishes the exposure period, it will transfer the $x[t]$ variable to the four slaves and will wait until all the matrix multiplications and additions are finalized. The master will then read the results and perform all of the remaining operations.

Figure 4.20b shows a possible way of arranging the 5-core clusters in order to form a large format image. In this case, the clusters at the edges of the array have a different arrangement of cores because of the geometric constraints. Two cores cannot be used and are represented by black squares. It must be noted that the current setup can be extended so that the master core also performs the computations for the timestamps from its corresponding slave cores, essentially creating a uniform LSTM imager. However, the size of the LSTM cell must be reduced in order to free up the RAM required to store the four extra c and h variables.

The final implementation uses 92% of the core instruction memory and has an execu-



(a) (b)

Figure 4.20: Clustering of *Ultra*Phase cores for LSTM computation (a) and possible tiling of said cluster to create a large format array (b). Master cores identified with a dot. Missing cores shown as black squares.

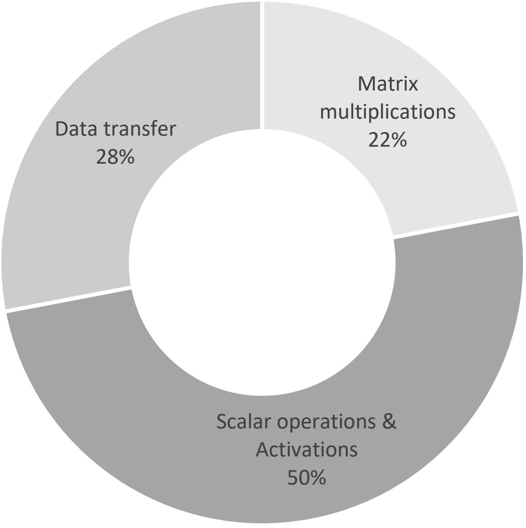


Figure 4.21: LSTM execution time when running on *Ultra*Phase. The scalar operations and activations take up the majority of the execution time because they run sequentially on the master core. In contrast, the matrix multiplications are distributed to the 4 slave cores and require significantly less processing time.

tion time of approximately 600 µs per cycle for each cluster of cores. The execution time will not change at larger format images because of the high level of parallelization. Figure 4.21 shows a breakdown of the execution time. The scalar operations and activations take up 50% of the total time because they run entirely on a single core, sequentially. Conversely, the matrix multiplications take the least amount of cycles because the operations are distributed across the four slave cores.

The setup used for the experiment is shown in Figure 4.22. At the beginning of every cycle, a synchronization pulse is emitted by the master core and used to trigger a 780 nm 70 ps FWHM PicoQuant VisIR laser [159]. A target is placed at a known distance from a PCB with a SPAD detector characterized in [160]. The output from the SPAD is connected to one of the inputs that are directly routed inside the SoC to the master core.

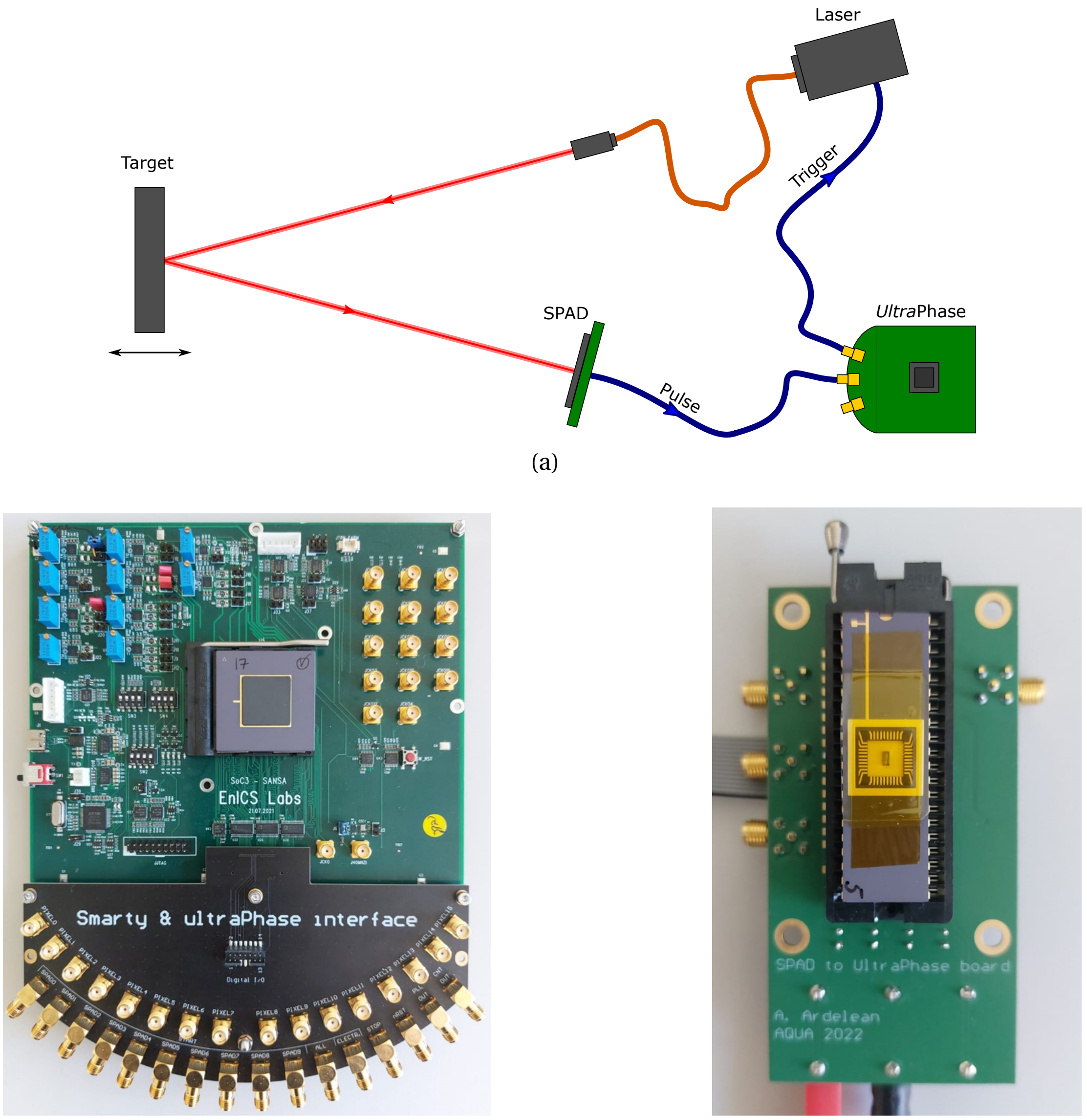
The data required to train the LSTM can also be acquired with *Ultra*Phase in the same setup, eliminating the need for simulated data or the errors introduced by using training data obtained under different conditions. The master core (the central core from Figure 4.20a) is programmed to run a loop that reads the output from the timing module and stores it into RAM. After 128 cycles, when the memory is full, the chip is read out and the process restarts until enough data has been collected.

4.7 Conclusion

*Ultra*Phase, a 12 × 24 SPAD imager in 180 nm and 16 nm 3D stacked technology was designed and the bottom tier fully characterized. The chip consists of a 3 × 6 grid of independent 32 bit processors that each interface with 16 SPAD pixels and can share data and synchronization signals with each other. The pixel outputs are connected to a reconfigurable combinational logic circuit that can implement a wide range of functions and preprocess the data before it reaches the main processing core. A separate path connects the pixels to a reconfigurable timing block that can act as a TDC and measure pulse widths and delays. Each core can execute programs of up to 256 instructions in length and has access to 4.096 kbit of scratchpad RAM.

Each core can execute 140 MOPS, beating the state of the art by an order of magnitude, at a maximum power consumption of 94.6 GOPS/W. The system offers the widest range of instructions compared to the other available alternatives and is a fully scalable architecture that does not require a control processor to operate.

Table 4.10 compares *Ultra*Phase with other state of the art general-purpose com-



(a)

(b)

(c)

Figure 4.22: *Ultra*Phase LSTM setup (a), a picture of the main board (b) and the PCB containing the standalone SPAD.

putational imagers.

Table 4.10: *Ultra*Phase state of the art comparison

	***Ultra*Phase**	**[48]**	**[49]**	**[46]**
Format	12 × 24	256 × 256	256 × 256	176 × 144
Pixel pitch	28.5 µm	32.26 µm	10 µm	33.6 µm
Fill factor	29.5%	6.2%	60%	3.19%
Detector type	SPAD	PD	PD	PD
Pixel processing cores	18	65536	4160	25344
Processing type	32 bit	analog	4096 × 1 bit 64 × 8 bit	analog
Controller processor	-	-	1 × 32 bit dual core	1 × 32 bit
Pixels per core	16	1	-	1
Technology	180 nm/16 nm	180 nm	180 nm	N/A
Performance[a]	140 MOPS	10 MOPS	2.8 MOPS	N/A
Power[a]	94.6 GOPS/W	525.5 GOPS/W	19 GOPS/W	<2 W[b]
RAM	4.096 kbit/core	13 bit/core 7 analog/core	64 bit/core	N/A
ROM	6.144 kbit/core	79 bit	N/A	17.9 kbit

[a] Per core.

[b] Total per chip, at 10 kfps acquiring and reading out images to external memory.

5 Conclusion

5.1 Summary

Vision systems built around conventional image sensors need to read, encode and transmit a large quantity of redundant pixel information that is then processed offline to extract features of interest. The constant evolution seen by CMOS image sensors that has resulted in multi-megapixel imagers with output data rates in the order of Gpx/s has made it necessary to develop a new type of sensor - the computational imager. These new architectures contain processing electronics that manipulate the raw pixel data and implement various image preprocessing algorithms to reduce the amount of redundant information that needs to be read from the sensor.

The emergence of large format single-photon SPAD-based imagers has emphasized the need for on-chip processing even more. If one wants to take advantage of the full capabilities of such a detector, a large amount of data needs to be handled in a short amount of time, with 4-year old designs already requiring output data rates of more than 3 GB/s, pushing the limit of current interfaces.

The aim of this thesis was to develop sensor architectures for computational imaging that overcome current limitations of SPAD imagers and can operate at high frame rates but with manageable output data rates. Multiple approaches were taken, for both 2D and 3D-stacked technologies, in mature 180 nm and modern 16 nm FinFET nodes, with various degrees of data processing parallelization and architecture customization. The concepts were implemented in 3 sensor architectures.

The first sensor, *kilo*Phase, is designed in a mature 2D technology and contains a novel token-based pixel readout scheme that ignores dark pixels and increases the system frame rate. The binary data collected from the SPAD array is accumulated in 2 separate accumulator banks that allow dead time free system operation. A group of computational units can simultaneously process the data from an entire array column and compute 2 MAC and one accumulate operation for each pixel. The architecture is very well suited for gated-FLIM as it can achieve a 4.38 ns gate and can handle up to 128 10 bit gate bins without the need for chip readout. When operating as a 10 bit intensity imager, the chip can achieve 227 fps but it is possible to extend the bit depth to 17 bit and increase the frame rate even further by using the computational units for accumulations. The output data rate of the detector is less than 10 Mb/s and despite the large amount of digital processing elements, the chip consumes less than 38 mW in normal operation.

The second design, *Mega*Phase is a megapixel SPAD imager fabricated in a 3D-stacked BSI technology and is an evolution of *kilo*Phase into a massively parallel and scalable architecture. The system consists of 16384 cores that each handle 64 SPADs and can simultaneously perform 2 MAC and one accumulate operation. Each group of 64 SPADs can be binned with multiple granularities to act as a pixel with non-unitary dynamic range, reducing the exposure time needed to obtain an image. The cores can operate in multiple modes, at different resolutions or processing operations and can be reconfigured from one frame to the next, without the need to stop the acquisition. The imager can support non-uniform resolution and operating modes, thus optimizing the amount of output data. Comparisons to conventional large format SPAD imagers show an increase in frame rate of up to 170× when operating as an intensity imager and 56× when in gated-FLIM mode. In addition, the output data rate is reduced by up to 3 orders of magnitude.

The final design, *Ultra*Phase is the first fully reconfigurable SPAD processing architecture and consists of 18 independent processors connected in a grid, each capable of running at 140 MOPS with a power consumption of less than 94.6 GOPS/W. The chip is a 3D-stacked FSI imager, with the bottom tier containing the processing electronics developed in 16 nm FinFET and the top tier with the SPAD and pixel circuitry in 180 nm CMOS. Each processor is connected to 16 SPADs and contains a reconfigurable front end that can preprocess the pixel data by implementing a wide range of combinational functions and a timing module that can be configured to measure counts, pulse widths or pulse delays. The design is fully customisable through software and can reconfigure itself at runtime. A custom assembly language is used to write programs of up to 256 instructions in length that can then be independently run by the processing cores. Two algorithms were ported to the design, showing both frame rate acceleration through parallel processing and output data rate reduction through compression.

Through the design process of the three previously described sensors, various technologies were explored, from 180 nm 2D to 3D-stacked 45/22 nm and 180/16 nm, backside and front-side illuminated. A few observations can be made regarding the best suited approach for future computational imagers. Firstly, while the 16 nm node allows for a significant amount of computational power to be included on the chip, the economics of using such an expensive technology for large format imagers makes it less attractive. Secondly, mature technology nodes such as the 180 nm one used in *kilo*Phase tend to have well performing SPADs but the amount of digital logic that can be included is fairly limited. The best compromise seems to be the use of technologies such as the 22/45 nm used for *Mega*Phase or similar nodes such as 40 nm and 55 nm. For the latter case, good quality SPADs are now being reported [160] and foundries can now offer in-house 3D stacking procedures that are much more attractive than the use of third parties like in the *Ultra*Phase project.

5.2 Future work

Future research efforts are primarily focused on improving the performance of the architectures presented in this work.

The importance of core-to-core communication was emphasized in the applications implemented on *Ultra*Phase, which makes it appealing to find ways of integrating similar features in massively parallel, albeit more simple, architectures such as *Mega*Phase or even vector processing systems such as *kilo*Phase. In addition, more complex inter-core communication networks not limited to direct neighbors should be studied and implemented in order to reduce the cost of sharing data across multiple processing elements.

With the development of parallel hardware comes the need for the development of software tools that can be used to program it. Assemblers and/or compilers that can take advantage of the parallel resources offered by computational imaging architectures such as *Ultra*Phase are a must if complex algorithms need to be implemented in a reasonable amount of time and with little to no errors.

Methods for improving detector sensitivity, such as the use of microlenses, can benefit from more work, as there is currently a discrepancy between designed and measured performance. In addition, the continuous shrinking of pixel pitch has made microlenses less effective, and new techniques are needed either for their manufacturing or their design.

Appendix

Superluminal motion-assisted 4D light-in-flight imaging

The work summarized in this chapter was published in [161]. Apart from participating in the development of the method, I contributed to the creation and validation of the mathematical model.

Advances in high-speed imaging techniques has allowed the capturing of ultrafast phenomena such as light propagation in air or other media. While recording light-in-flight in 3D *xyt* coordinates has already been reported, reconstructing the fourth z coordinate has been difficult. The authors demonstrated the four dimensional light-in-flight imaging based on observation of superluminal motion captured with a large format SPAD imager.

The experimental setup consisted of a picosecond laser and SPAD camera [66] controlled by a pulse generator. A set of mirrors confined the light path inside an acrylic box filled with artificial fog to create a scattering medium. The optical center of the camera lens was considered as the origin of the Carthesian coordinate system. Figure 5.1 shows the experimental setup.

As light propagates between the mirrors at a finite speed, scattered photons from each point along the path reach the camera after an amount of time dependent on the distance between said point and the camera. As a result, depending on the position of the camera, light may appear to travel faster in some points and slower in others. In some cases, the light appears to travel at superluminal speeds, a phenomenon already observed in astronomy. The apparent velocity can then be used to estimate the light propagation vector in the 3D Carthesian space.

Figure 5.2a is a color-coded plot of observation time of the light path. Red points are observed earlier and blue later. The time data needs to be subdivided into straight paths so that the corresponding vectors can be extracted. A 2D Gaussian mixture

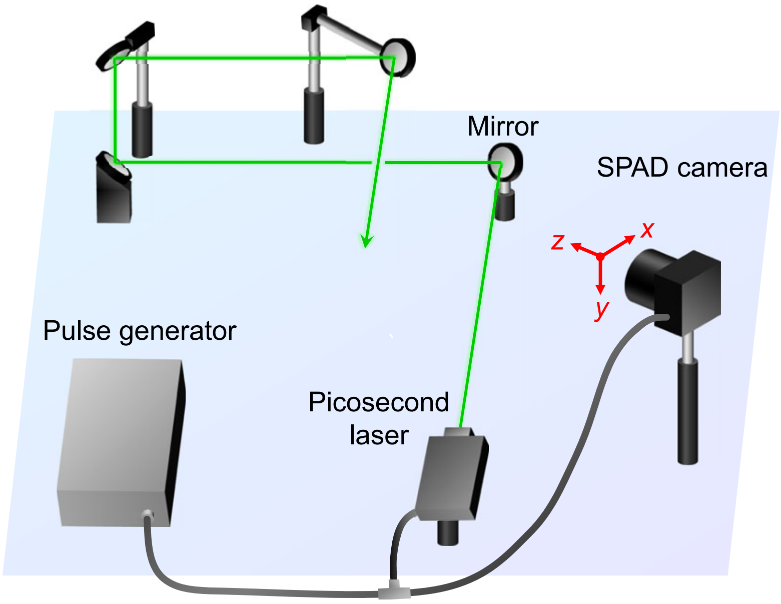
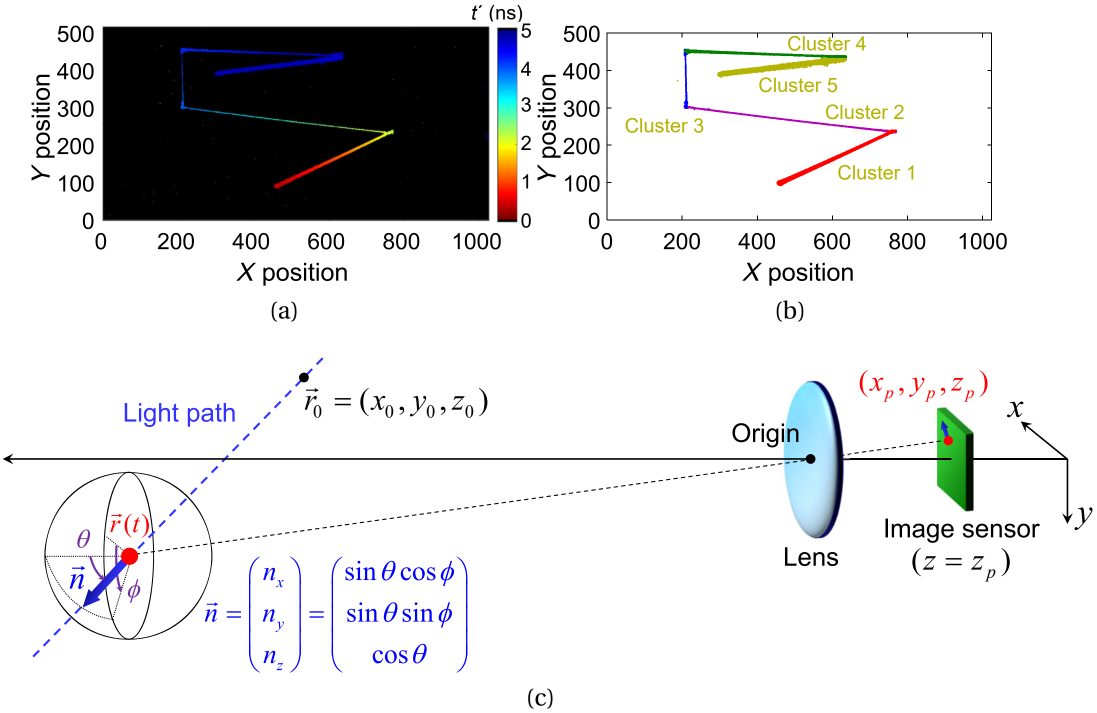


Figure 5.1: The experimental setup used for light-in-flight imaging. A picosecond laser and a SPAD camera are controlled by a pulse generator. The scene consists of a light path confined by a set of mirrors inside an acrylic box (not shown) filled with artificial fog [161].

model was used to fit the data for clustering, a common unsupervised machine-learning technique. Figure 5.2b shows the separated data points according to their *xy* coordinates. Figure 5.2c indicates the coordinate system for the analysis that was used to derive a theoretical formula for performing least-square regression to estimate the light trajectory. Finally, figure 5.3 shows the 4D point cloud reconstructed from the measured data without any prior knowledge. The results are in good accordance with the ground truth.



(a) (b)

(c)

Figure 5.2: (a) Color-coded plot of observation times along the propagation path, with red being observed earlier; (b) Light path divided into clusters using a 2D Gaussian mixture model; (c)Coordinate system used for light-in-flight reconstruction [161].

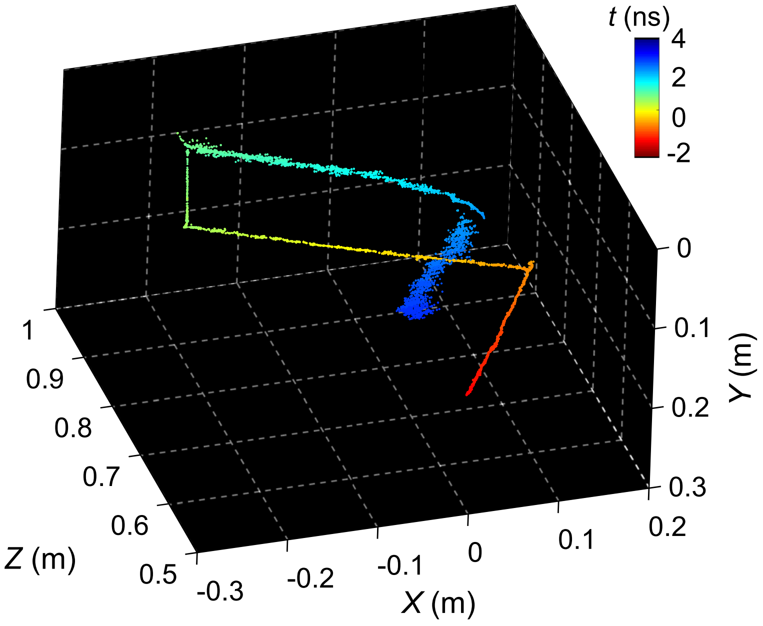


Figure 5.3: Reconstructed 4D light-in-flight path represented as a point cloud in 3D Carthesian coordinates with color denoting the time coordinate [161].

Bibliography

[1] P. Pande and J. A. Jo, "Automated Analysis of Fluorescence Lifetime Imaging Microscopy (FLIM) Data Based on the Laguerre Deconvolution Method", *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 172–181, 2011. DOI: 10.1109/TBME.2010.2084086.

[2] Y. Won and S. Lee, "Real-Time Confocal Fluorescence Lifetime Imaging Microscopy (FLIM) for Spectroscopic Sensing of Cancer Tissues", in *2018 Conference on Precision Electromagnetic Measurements (CPEM 2018)*, 2018, pp. 1–2. DOI: 10.1109/CPEM.2018.8500984.

[3] K. Suhling, P.-H. Chung, and J. Levitt, "Advances in time-resolved fluorescence microscopy: Simultaneous FRAP, FLIM and tr-FAIM to image rotational and translation diffusion in living cells", in *2011 Functional Optical Imaging*, 2011, pp. 1–1. DOI: 10.1109/FOI.2011.6154838.

[4] E. Vargas, J. N. P. Martel, G. Wetzstein, and H. Arguello, *Time-Multiplexed Coded Aperture Imaging: Learned Coded Aperture and Pixel Exposures for Compressive Imaging Systems*, 2021. DOI: 10.48550/ARXIV.2104.02820. [Online]. Available: https://arxiv.org/abs/2104.02820.

[5] A. Yu, T. Jiang, W. Chen, and X. Tan, "A hyperspectral image fusion algorithm based on Compressive Sensing", in *2012 4th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2012, pp. 1–4.

[6] V. Duran, F. Soldevila, E. Irles, P. Clemente, E. Tajahuerce, P. Andres, and J. Lancis, "Compressive imaging in scattering media", *Opt. Express*, vol. 23, no. 11, pp. 14 424–14 433, 2015. DOI: 10.1364/OE.23.014424.

[7] R. R. Deshpande, M. R. Bhatt, and C. H. R. Madhavi, "Accuracy in Depth Recovery and 3D Image Synthesis From Single Image Using Multi-Color Filter Aperture and Shallow Depth of Field", *IEEE Access*, vol. 9, pp. 123 528–123 540, 2021. DOI: 10.1109/ACCESS.2021.3109865.

[8] X. Yang, J. Sun, and W. Diao, “Depth Image Inpainting for RGB-D Camera Based on Light Field EPI”, in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, 2018, pp. 214–219. DOI: 10.1109/ICIVC.2018.8492912.

[9] L. Zhao, J. Liang, H. Bai, A. Wang, and Y. Zhao, “Convolutional neural network-based depth image artifact removal”, in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 2438–2442. DOI: 10.1109/ICIP.2017.8296720.

[10] K. Uruma, K. Konishi, T. Takahashi, and T. Furukawa, “High resolution depth image recovery algorithm based on the modeling of the sum of an average distance image and a surface image”, in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 2836–2840. DOI: 10.1109/ICIP.2016.7532877.

[11] W. Chantara and Y.-S. Ho, “Initial depth estimation using adaptive window size with light field image”, in *2018 International Workshop on Advanced Image Technology (IWAIT)*, 2018, pp. 1–3. DOI: 10.1109/IWAIT.2018.8369722.

[12] N. Diaz, J. Ramirez, E. Vera, and H. Arguello, “Adaptive Multisensor Acquisition via Spatial Contextual Information for Compressive Spectral Image Classification”, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 9254–9266, 2021. DOI: 10.1109/JSTARS.2021.3111508.

[13] C. V. Correa, H. Arguello, and G. R. Arce, “Spatio-spectral uniform multi-frame coded apertures for compressive spectral imaging”, in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2015, pp. 614–618. DOI: 10.1109/GlobalSIP.2015.7418269.

[14] L. Huang, R. Luo, X. Liu, and X. Hao, “Spectral imaging with deep learning”, *Light: Science and Applications*, vol. 11, no. 61, 2022. DOI: 10.1038/s41377-022-00743-6.

[15] W. Chantara and Y.-S. Ho, “Initial depth estimation using adaptive window size with light field image”, in *2018 International Workshop on Advanced Image Technology (IWAIT)*, 2018, pp. 1–3. DOI: 10.1109/IWAIT.2018.8369722.

[16] M. Lamba, K. K. Rachavarapu, and K. Mitra, “Harnessing Multi-View Perspective of Light Fields for Low-Light Imaging”, *IEEE Transactions on Image Processing*, vol. 30, pp. 1501–1513, 2021. DOI: 10.1109/TIP.2020.3045617.

[17] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu, “Light Field Image Processing: An Overview”, *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 7, pp. 926–954, 2017. DOI: 10.1109/JSTSP.2017.2747126.

[18] G. H. Jin, G. R. Choi, H. H. Park, T. S. Lee, and S. B. Lee, “Defining gross tumor volume using positron emission tomography/computed tomography phantom studies”, in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, pp. 2473–2476. DOI: 10.1109/EMBC.2013.6610041.

[19] Y. Kuang, G. Pratx, M. Bazalova, B. Meng, J. Qian, and L. Xing, “First Demonstration of Multiplexed X-Ray Fluorescence Computed Tomography (XFCT) Imaging”, *IEEE Transactions on Medical Imaging*, vol. 32, no. 2, pp. 262–267, 2013. DOI: 10.1109/TMI.2012.2223709.

[20] P. J. L. Riviere, P. Vargas, G. Fu, and L. J. Meng, “Accelerating X-ray fluorescence computed tomography”, in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2009, pp. 1000–1003. DOI: 10.1109/IEMBS.2009.5333568.

[21] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, “Image and Depth from a Conventional Camera with a Coded Aperture”, *ACM Trans. Graph.*, vol. 26, no. 3, 70–es, 2007. DOI: 10.1145/1276377.1276464.

[22] V. Paramonov, I. Panchenko, V. Bucha, A. Drogolyub, and S. Zagoruyko, “Depth Camera Based on Color-Coded Aperture”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, pp. 910–918. DOI: 10.1109/CVPRW.2016.118.

[23] Y. Sekikawa, S.-w. Leigh, and K. Suzuki, “Coded Lens: Using Coded Aperture for Low-Cost and Versatile Imaging”, in *ACM SIGGRAPH 2014 Posters*, ser. SIGGRAPH ’14, Vancouver, Canada: Association for Computing Machinery, 2014, ISBN: 9781450329583. DOI: 10.1145/2614217.2614227.

[24] Y. Hirano, A. Matsuda, and Y. Hiraoka, “Recent advancements in structured-illumination microscopy toward live-cell imaging”, *Microscopy*, vol. 64, no. 4, pp. 237–249, 2015. DOI: 10.1093/jmicro/dfv034.

[25] M. J. Huttunen, A. Abbas, J. Upham, and R. W. Boyd, “Label-Free Super-Resolution Microscopy with Coherent Nonlinear Structured-Illumination”, in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, 2018, pp. 1–4. DOI: 10.1109/ICTON.2018.8473971.

[26] Y. Terui, “Image processing for structured illumination microscopy”, in *2015 14th Workshop on Information Optics (WIO)*, 2015, pp. 1–3. DOI: 10.1109/WIO.2015.7206919.

[27] L. Zhao, C. Han, Y. Shu, M. Lv, Y. Liu, T. Zhou, Z. Yan, and X. Liu, “Improved Imaging Performance in Super-Resolution Localization Microscopy by YALL1 Method”, *IEEE Access*, vol. 6, pp. 5438–5446, 2018. DOI: 10.1109/ACCESS.2018.2793847.

[28] T.-a. Pham, E. Soubies, D. Sage, and M. Unser, “Closed-Form Expression Of The Fourier Ring-Correlation For Single-Molecule Localization Microscopy”, in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 2019, pp. 321–324. DOI: 10.1109/ISBI.2019.8759279.

[29] A. Rodríguez-Vázquez, J. Fernández-Berni, J. A. Leñero-Bardallo, I. Vornicu, and R. Carmona-Galán, “CMOS Vision Sensors: Embedding Computer Vision at Imaging Front-Ends”, *IEEE Circuits and Systems Magazine*, vol. 18, no. 2, pp. 90–107, 2018. DOI: 10.1109/MCAS.2018.2821772.

[30] *CHR71000*, 2022. [Online]. Available: https://ams.com/en/chr71000.

[31] *GJ01611*, 2022. [Online]. Available: https://www.gigajot.tech/qis.

[32] I. Takayanagi and J. Nakamura, “High-Resolution CMOS Video Image Sensors”, *Proceedings of the IEEE*, vol. 101, no. 1, pp. 61–73, 2013. DOI: 10.1109/JPROC.2011.2178569.

[33] S. Kawahito, J.-H. Park, K. Isobe, S. Shafie, T. Iida, and T. Mizota, “A CMOS Image Sensor Integrating Column-Parallel Cyclic ADCs with On-Chip Digital Error Correction Circuits”, in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, 2008, pp. 56–595. DOI: 10.1109/ISSCC.2008.4523054.

[34] A. B. R. Vazquez, R. C. Galan, J. F. Berni, V. B. Sanchez, and J. A. L. Bardallo, “In the quest of vision-sensors-on-chip: Pre-processing sensors for data reduction”, in *Image Sensors and Imaging Systems*, 2017, pp. 96–101. DOI: 10.2352/ISSN.2470-1173.2017.11.IMSE-195.

[35] A. N. Belbachir, in *Smart Cameras*, Springer, 2009, pp. 312–347, ISBN: 9781441909527. DOI: 10.1007/978-1-4419-0953-4.

[36] A. Torralba, “How many pixels make an image?”, *Visual Neuroscience*, vol. 26, no. 1, pp. 123–131, 2009. DOI: 10.1017/S0952523808080930.

[37] C. Zhang, S. Lindner, I. M. Antolovic, M. Wolf, and E. Charbon, “A CMOS SPAD imager with collision detection and 128 dynamically reallocating TDCs for single-photon counting and 3D time-of-flight imaging”, *Sensors*, vol. 18, no. 11, 2018, ISSN: 1424-8220. DOI: 10.3390/s18114016. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/4016.

[38] R. K. Henderson, N. Johnston, S. W. Hutchings, I. Gyongy, T. A. Abbas, N. Dutton, M. Tyler, S. Chan, and J. Leach, “5.7 A 256×256 40nm/90nm CMOS 3D-Stacked 120dB Dynamic-Range Reconfigurable Time-Resolved SPAD Imager”, in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019, pp. 106–108. DOI: 10.1109/ISSCC.2019.8662355.

[39] S. W. Hutchings, N. Johnston, I. Gyongy, T. Al Abbas, N. A. W. Dutton, M. Tyler, S. Chan, J. Leach, and R. K. Henderson, “A Reconfigurable 3-D-Stacked SPAD Imager With In-Pixel Histogramming for Flash LIDAR or High-Speed Time-of-Flight Imaging”, *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 2947–2956, 2019. DOI: 10.1109/JSSC.2019.2939083.

[40] F. Mattioli Della Rocca, H. Mai, S. W. Hutchings, T. A. Abbas, K. Buckbee, A. Tsiamis, P. Lomax, I. Gyongy, N. A. W. Dutton, and R. K. Henderson, “A 128 × 128 spad motion-triggered time-of-flight image sensor with in-pixel histogram and column-parallel vision processor”, *IEEE Journal of Solid-State Circuits*, vol. 55, no. 7, pp. 1762–1775, 2020. DOI: 10.1109/JSSC.2020.2993722.

[41] M. Suarez, V. M. Brea, J. Fernandez-Berni, R. Carmona-Galan, D. Cabello, and A. Rodriguez-Vazquez, “Low-Power CMOS Vision Sensor for Gaussian Pyramid Extraction”, *IEEE Journal of Solid-State Circuits*, vol. 52, no. 2, pp. 483–495, 2017. DOI: 10.1109/JSSC.2016.2610580.

[42] L. Nielsen, M. Mahowald, and C. Mead, *SeeHear* (Technical Reports TFRT-7355), English. Department of Automatic Control, Lund Institute of Technology (LTH), 1987.

[43] J. E. Tanner and C. Mead, “A Correlating Optical Motion Detector”, in *Proceedings, Conference on Advanced Research in VLSI*, 1984, ISBN: 089006136X.

[44] M. A. Mahowald and C. Mead, “The Silicon Retina”, *Scientific American*, vol. 264, no. 5, pp. 76–83, 1991, ISSN: 00368733, 19467087.

[45] C. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley Publishing Company, 1989, ISBN: 0201059924.

[46] teledyne e2v.com, *Eye-RIS VSoC*, 2022. [Online]. Available: https://imaging.teledyne-e2v.com/content/uploads/2019/01/33613-AnaFocus_Eye-RIS-VSoC_v3_AW_WEB.pdf.

[47] G. Linan, R. Dominguez-Castro, S. Espejo, and A. Rodriguez-Vazquez, “ACE16K: An advanced focal-plane analog programmable array processor”, in *Proceedings of the 27th European Solid-State Circuits Conference*, 2001, pp. 201–204.

[48] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, “A 100,000 fps vision sensor with embedded 535GOPS/W 256×256 SIMD processor array”, in *2013 Symposium on VLSI Circuits*, 2013, pp. C182–C183.

[49] C. Shi, J. Yang, Y. Han, Z. Cao, Q. Qin, L. Liu, N.-J. Wu, and Z. Wang, “A 1000 fps Vision Chip Based on a Dynamically Reconfigurable Hybrid Architecture Comprising a PE Array Processor and Self-Organizing Map Neural Network”, *IEEE Journal of Solid-State Circuits*, vol. 49, no. 9, pp. 2067–2082, 2014. DOI: 10.1109/JSSC.2014.2332134.

[50] J. Poikonen, M. Laiho, and A. Paasio, “MIPA4k: A 64×64 cell mixed-mode image processor array”, in *2009 IEEE International Symposium on Circuits and Systems*, 2009, pp. 1927–1930. DOI: 10.1109/ISCAS.2009.5118161.

[51] A. Rodriguez-Vazquez, R. Dominguez-Castro, F. Jimenez-Garrido, S. Morillas, A. Garcia, C. Utrera, M. D. Pardo, J. Listan, and R. Romay, “A CMOS Vision System On-Chip with Multi-Core, Cellular Sensory-Processing Front-End”, in *Cellular Nanoscale Sensory Wave Computing*, C. Baatar, W. Porod, and T. Roska, Eds. Boston, MA: Springer US, 2010, pp. 129–146, ISBN: 9781441910110. DOI: 10.1007/978-1-4419-1011-0_6.

[52] A. Zarandy, C. Rekeczky, P. Foldesy, R. Carmona-Galan, G. L. Cembrano, S. Gergely, A. Rodriguez-Vazquez, and T. Roska, “VISCUBE: A Multi-Layer Vision Chip”, in *Focal-Plane Sensor-Processor Chips*, A. Zarandy, Ed. New York, NY: Springer New York, 2011, pp. 181–208, ISBN: 9781441964755. DOI: 10.1007/978-1-4419-6475-5_8.

[53] P. Foldesy, A. Zarandy, C. Rekeczky, and T. Roska, “High performance processor array for image processing”, in *2007 IEEE International Symposium on Circuits and Systems*, 2007, pp. 1177–1180. DOI: 10.1109/ISCAS.2007.378260.

[54] A. Zarandy, Ed., *Focal-plane sensor-processor chips.* New York: Springer Science+Business Media BV, 2011.

[55] Y. Liu, L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, “High-speed Light-weight CNN Inference via Strided Convolutions on a Pixel Processor Array”, in *BMVC*, 2020.

[56] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, *Fully Embedding Fast Convolutional Networks on Pixel Processor Arrays*, 2020. DOI: 10.48550/ARXIV.2004.12525. [Online]. Available: https://arxiv.org/abs/2004.12525.

[57] A. Rochas, M. Gosch, A. Serov, P. Besse, R. Popovic, T. Lasser, and R. Rigler, “First fully integrated 2-d array of single-photon detectors in standard cmos technology”, *IEEE Photonics Technology Letters*, vol. 15, no. 7, pp. 963–965, 2003. DOI: 10.1109/LPT.2003.813387.

[58] C. Niclass, A. Rochas, P. Besse, and E. Charbon, "A cmos single photon avalanche diode array for 3d imaging", in *2004 IEEE International Solid-State Circuits Conference (IEEE Cat. No.04CH37519)*, 2004, 120–517 Vol.1. DOI: 10.1109/ISSCC.2004.1332623.

[59] C. Niclass, A. Rochas, P.-A. Besse, R. Popovic, and E. Charbon, "Cmos imager based on single photon avalanche diodes", in *The 13th International Conference on Solid-State Sensors, Actuators and Microsystems, 2005. Digest of Technical Papers. TRANSDUCERS '05.*, vol. 1, 2005, 1030–1034 Vol. 1. DOI: 10.1109/SENSOR.2005.1496631.

[60] C. Niclass, A. Rochas, P.-A. Besse, and E. Charbon, "Design and characterization of a cmos 3-d image sensor based on single photon avalanche diodes", *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1847–1854, 2005. DOI: 10.1109/JSSC.2005.848173.

[61] C. Niclass, M. Sergio, and E. Charbon, "A single photon avalanche diode array fabricated in deep-submicron cmos technology", in *Proceedings of the Design Automation and Test in Europe Conference*, vol. 1, 2006, pp. 1–6. DOI: 10.1109/DATE.2006.243987.

[62] C. Niclass, M. Sergio, and E. Charbon, "A cmos 64×48 single photon avalanche diode array with event-driven readout", in *2006 Proceedings of the 32nd European Solid-State Circuits Conference*, 2006, pp. 556–559. DOI: 10.1109/ESSCIR.2006.307485.

[63] J. Richardson, R. Walker, L. Grant, D. Stoppa, F. Borghetti, E. Charbon, M. Gersbach, and R. Henderson, "A 32×32 50ps Resolution 10 bit Time to Digital Converter Array in 130nm CMOS for Time Correlated Imaging", Sep. 2009, pp. 77–80. DOI: 10.1109/CICC.2009.5280890.

[64] P. Padmanabhan, C. Zhang, M. Cazzaniga, B. Efe, A. R. Ximenes, M.-J. Lee, and E. Charbon, "7.4 A 256×128 3D-Stacked (45nm) SPAD FLASH LiDAR with 7-Level Coincidence Detection and Progressive Gating for 100m Range and 10klux Background Light", in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 111–113. DOI: 10.1109/ISSCC42613.2021.9366010.

[65] I. Gyongy, A. T. Erdogan, N. A. W. Dutton, G. M. Martín, A. Gorman, H. Mai, F. M. Della Rocca, and R. K. Henderson, *A direct time-of-flight image sensor with in-pixel surface detection and dynamic vision*, 2022. DOI: 10.48550/ARXIV.2209.11772. [Online]. Available: https://arxiv.org/abs/2209.11772.

[66] K. Morimoto, A. Ardelean, M.-L. Wu, A. C. Ulku, I. M. Antolovic, C. Bruschini, and E. Charbon, “Megapixel time-gated SPAD image sensor for 2D and 3D imaging applications”, *Optica*, vol. 7, no. 4, pp. 346–354, 2020. DOI: 10.1364/OPTICA.386574.

[67] T. Okino, S. Yamada, Y. Sakata, S. Kasuga, M. Takemoto, Y. Nose, H. Koshida, M. Tamaru, Y. Sugiura, S. Saito, S. Koyama, M. Mori, Y. Hirose, M. Sawada, A. Odagawa, and T. Tanaka, “5.2 A 1200×900 6μm 450fps Geiger-Mode Vertical Avalanche Photodiodes CMOS Image Sensor for a 250m Time-of-Flight Ranging System Using Direct-Indirect-Mixed Frame Synthesis with Configurable-Depth-Resolution Down to 10cm”, in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 96–98. DOI: 10.1109/ISSCC19947.2020.9063045.

[68] C. Inc., *3.2-megapixel SPAD sensor*, 2022. [Online]. Available: https://global.canon/en/news/2021/20211215.html.

[69] F. Severini, I. Cusini, D. Berretta, K. Pasquinelli, A. Incoronato, and F. Villa, “SPAD Pixel With Sub-NS Dead-Time for High-Count Rate Applications”, *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 28, no. 2: Optical Detectors, pp. 1–8, 2022. DOI: 10.1109/JSTQE.2021.3124825.

[70] P. Keshavarzian, F. Gramuglia, E. Kizilkan, C. Bruschini, S. S. Tan, M. Tng, D. Chong, E. Quek, M.-J. Lee, and E. Charbon, “Low-noise high-dynamic-range single-photon avalanche diodes with integrated PQAR circuit in a standard 55nm BCD process”, in *Advanced Photon Counting Techniques XVI*, M. A. Itzler, J. C. Bienfang, and K. A. McIntosh, Eds., International Society for Optics and Photonics, vol. 12089, SPIE, 2022, 120890B. DOI: 10.1117/12.2618349.

[71] A. C. Ulku, C. Bruschini, I. M. Antolović, Y. Kuo, R. Ankri, S. Weiss, X. Michalet, and E. Charbon, “A 512 × 512 SPAD image sensor with integrated gating for widefield FLIM”, *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 25, no. 1, pp. 1–12, 2019. DOI: 10.1109/JSTQE.2018.2867439.

[72] M. Y. Berezin and S. Achilefu, “Fluorescence Lifetime Measurements and Biological Imaging”, *Chemical Reviews*, vol. 110, no. 5, pp. 2641–2684, 2010, PMID: 20356094. DOI: 10.1021/cr900343z.

[73] C. Bruschini, H. Homulle, I. M. Antolovic, S. Burri, and E. Charbon, “Single-photon avalanche diode imagers in biophotonics: review and outlook”, *Light: Science and Applications*, vol. 8, p. 87, 2019. DOI: 10.1038/s41377-019-0191-5.

[74] L. Marcu, “Fluorescence Lifetime Techniques in Medical Applications”, *Annals of Biomedical Engineering*, vol. 40, no. 2, pp. 304–331, 2012.

[75] X. Liu, D. Lin, W. Becker, J. Niu, B. Yu, L. Liu, and J. Qu, "Fast fluorescence lifetime imaging techniques: A review on challenge and development", *Journal of Innovative Optical Health Sciences*, vol. 12, no. 5, p. 1 930 003, 2019.

[76] R. K. Henderson, B. R. Rae, and D.-U Li, "Complementary metal-oxide-semiconductor (CMOS) sensors for fluorescence lifetime imaging (FLIM)", in *High Performance Silicon Imaging*, Elsevier, 2014, pp. 312–347, ISBN: 9780857095985. DOI: 10.1533/9780857097521.2.312.

[77] L. M. Hirvonen and K. Suhling, "Fast Timing Techniques in FLIM Applications", *Frontiers in Physics*, vol. 8, 2020, ISSN: 2296-424X. DOI: 10.3389/fphy.2020.00161.

[78] S. Rajoria, L. Zhao, X. Intes, and M. Barroso, "FLIM-FRET for Cancer Applications", *Current Molecular Imaging*, vol. 3, no. 2, pp. 144–161, 2014.

[79] J. W. Borst and A. J. W. G. Visser, "Fluorescence lifetime imaging microscopy in life sciences", *Measurement Science and Technology*, vol. 21, no. 10, p. 102 002, 2010. DOI: 10.1088/0957-0233/21/10/102002.

[80] K. Suhling, P. M. W. French, and D. Phillips, "Time-resolved fluorescence microscopy", *Photochem. Photobiol. Sci.*, vol. 4, pp. 13–22, 1 2005. DOI: 10.1039/B412924P.

[81] L. M. Hirvonen and K. Suhling, "Wide-field TCSPC: methods and applications", *Measurement Science and Technology*, vol. 28, no. 1, p. 012 003, 2016. DOI: 10.1088/1361-6501/28/1/012003.

[82] W. Becker, A. Bergmann, M. A. Hink, K. Konig, K. Benndorf, and C. Biskup, "Fluorescence lifetime imaging by time-correlated single-photon counting", *Microscopy research and technique*, vol. 63, no. 1, pp. 58–66, 2004. DOI: 10.1002/jemt.10421.

[83] picoquant.com, *PicoQuant*, 2022. [Online]. Available: https://www.picoquant.com/products/category/tcspc-and-time-tagging-modules.

[84] A. C. Ulku, *Large-Format Time-Gated SPAD Cameras for Real-Time Phasor-Based FLIM*, 2021. [Online]. Available: https://infoscience.epfl.ch/record/285478?ln=en.

[85] M. Maus, M. Cotlet, J. Hofkens, T. Gensch, F. C. De Schryver, J. Schaffer, and C. A. Seidel, "An experimental comparison of the maximum likelihood estimation and nonlinear least-squares fluorescence lifetime analysis of single molecules", *Analytical chemistry*, vol. 73, no. 9, pp. 2078–2086, 2001. DOI: 10.1021/ac000877g.

[86] P. Hall and B. Selinger, “Better estimates of exponential decay parameters”, *The Journal of Physical Chemistry*, vol. 85, no. 20, pp. 2941–2946, 1981. DOI: 10.1021/j150620a019.

[87] G. Wu, T. Nowotny, Y. Chen, and D. D.-U. Li, “GPU acceleration of time-domain fluorescence lifetime imaging”, *Journal of Biomedical Optics*, vol. 21, no. 1, p. 017 001, 2016. DOI: 10.1117/1.JBO.21.1.017001.

[88] J. Jo, Q. Fang, and L. Marcu, “Ultrafast method for the analysis of fluorescence lifetime imaging microscopy data based on the Laguerre expansion technique”, *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 11, no. 4, pp. 835–845, 2005. DOI: 10.1109/JSTQE.2005.857685.

[89] K. K. Sharman, A. Periasamy, H. Ashworth, and J. N. Demas, “Error Analysis of the Rapid Lifetime Determination Method for Double-Exponential Decays and New Windowing Schemes”, *Analytical Chemistry*, vol. 71, no. 5, pp. 947–952, 1999. DOI: 10.1021/ac981050d.

[90] R. M. Ballew and J. N. Demas, “An error analysis of the rapid lifetime determination method for the evaluation of single exponential decays”, *Analytical Chemistry*, vol. 61, no. 1, pp. 30–33, 1989. DOI: 10.1021/ac00176a007.

[91] A. Vard, S. F. Silva, J. P. Domingues, and A. M. Morgado, “Accurate Rapid Lifetime Determination on Time-Gated FLIM Microscopy with Optical Sectioning”, *Journal of Healthcare Engineering*, vol. 2018, 2018. DOI: 10.1155/2018/1371386.

[92] D. D. U. Li, R. Andrews, J. Arlt, and R. Henderson, “Hardware implementation algorithm and error analysis of high-speed fluorescence lifetime sensing systems using center-of-mass method”, *Journal of Biomedical Optics*, vol. 15, no. 1, p. 017 006, 2010. DOI: 10.1117/1.3309737.

[93] D. D. U. Li, H. Yu, and Y. Chen, “Fast bi-exponential fluorescence lifetime imaging analysis methods”, *Optics Letters*, vol. 40, no. 3, pp. 336–339, 2015. DOI: 10.1364/OL.40.000336.

[94] S. P. Poland, A. T. Erdogan, N. Krstajić, J. Levitt, V. Devauges, R. J. Walker, D. D.-U. Li, S. M. Ameer-Beg, and R. K. Henderson, “New high-speed centre of mass method incorporating background subtraction for accurate determination of fluorescence lifetime”, *Opt. Express*, vol. 24, no. 7, pp. 6899–6915, 2016. DOI: 10.1364/OE.24.006899.

[95] M. A. Digman, V. R. Caiolfa, M. Zamai, and E. Gratton, “The phasor approach to fluorescence lifetime imaging analysis”, *Biophysical Journal*, vol. 94, no. 2, pp. L14–L16, 2008. DOI: 10.1529/biophysj.107.120154.

[96] *SimFCS*, 2022. [Online]. Available: https://www.lfd.uci.edu/globals/.

[97] C. Stringari, A. Cinquin, O. Cinquin, M. A. Digman, P. J. Donovan, and E. Gratton, “Phasor approach to fluorescence lifetime microscopy distinguishes different metabolic states of germ cells in a live tissue”, *Proceedings of the National Academy of Sciences*, vol. 108, no. 33, pp. 13 582–13 587, 2011. DOI: 10.1073/pnas.1108161108.

[98] A. Celli, S. Sanchez, T. Behne M. abd Hazlett, E. Gratton, and T. Mauro, “The epidermal Ca2+ gradient: measurement using the phasor representation of fluorescent lifetime imaging”, *Byophisics Journal*, vol. 98, no. 5, pp. 911–921, 2010. DOI: 10.1016/j.bpj.2009.10.055.

[99] A. Ulku, A. Ardelean, M. Antolovic, S. Weiss, E. Charbon, C. Bruschini, and X. Michalet, “Wide-field time-gated SPAD imager for phasor-based FLIM applications”, *Methods and Applications in Fluorescence*, vol. 8, no. 2, p. 024 002, 2020. DOI: 10.1088/2050-6120/ab6ed7.

[100] X. Michalet, “Continuous and discrete phasor analysis of binned or time-gated periodic decays”, *AIP Advances*, vol. 11, p. 035 331, 2021. DOI: 10.1088/2050-6120/ab6ed7.

[101] M.-J. Lee and E. Charbon, “Progress in single-photon avalanche diode image sensors in standard CMOS: From two-dimensional monolithic to three-dimensional-stacked technology”, *Japanese journal of applied physics*, vol. 57, no. 10, 1002A3, 2018.

[102] P. W. R. Connolly, X. Ren, R. K. Henderson, and G. S. Buller, “Hot pixel classification of single-photon avalanche diode detector arrays using a log-normal statistical distribution”, *Electronics letters*, vol. 55, no. 18, pp. 1004–1006, 2019. DOI: 10.1049/el.2019.1427.

[103] M. W. Fishburn, *Fundamentals of CMOS Single-Photon Avalanche Diodes*, 2012. [Online]. Available: https://doi.org/10.4233/uuid:7ed6e57d-404e-4372-8053-6b0b5c7fa0fe.

[104] I. M. Antolovic, S. Burri, C. Bruschini, R. Hoebe, and E. Charbon, “Nonuniformity analysis of a 65-kpixel CMOS SPAD imager”, *IEEE Transactions on Electron Devices*, vol. 63, no. 1, pp. 57–64, 2016. DOI: 10.1109/TED.2015.2458295.

[105] I. Gyongy, N. Calder, A. Davies, N. A. W. Dutton, R. R. Duncan, C. Rickman, P. Dalgarno, and R. K. Henderson, “A 256 × 256 , 100-kfps, 61% fill-factor SPAD image sensor for time-resolved microscopy applications”, *IEEE Transactions on Electron Devices*, vol. 65, no. 2, pp. 547–554, 2018. DOI: 10.1109/TED.2017.2779790.

[106] P. Seitz and A. Theuwissen, Eds., *Single-photon imaging*. Berlin: Springer Berlin, Heidelberg, 2011, ISBN: 9783642184420.

[107] C. Veerappan, *Single-photon avalanche diodes for cancer diagnosis*, 2016. [Online]. Available: https://doi.org/10.4233/uuid:7db13e84-9ced-4c9c-94fa-8c2a14ad6679.

[108] M. W. Fishburn, *Fundamentals of CMOS single-photon avalanche diodes*, 2012. [Online]. Available: https://doi.org/10.4233/uuid:7ed6e57d-404e-4372-8053-6b0b5c7fa0fe.

[109] K. Morimoto and E. Charbon, “A Scaling Law for SPAD Pixel Miniaturization”, *Sensors*, vol. 21, no. 10, 2021, ISSN: 1424-8220. DOI: 10.3390/s21103447.

[110] J. M. Pavia, M. Wolf, and E. Charbon, “Measurement and modeling of microlenses fabricated on single-photon avalanche diode arrays for fill factor recovery”, *Opt. Express*, vol. 22, no. 4, pp. 4202–4213, 2014. DOI: 10.1364/OE.22.004202.

[111] I. Gyongy, A. Davies, B. Gallinet, N. A. Dutton, R. R. Duncan, C. Rickman, R. K. Henderson, and P. A. Dalgarno, “Cylindrical microlensing for enhanced collection efficiency of small pixel SPAD arrays in single-molecule localisation microscopy”, *Opt. Express*, vol. 26, no. 3, pp. 2280–2291, 2018. DOI: 10.1364/OE.26.002280.

[112] I. M. Antolovic, A. C. Ulku, E. Kizilkan, S. Lindner, F. Zanella, R. Ferrini, M. Schnieper, E. Charbon, and C. Bruschini, “Optical-stack optimization for improved SPAD photon detection efficiency”, in *Quantum Sensing and Nano Electronics and Photonics XVI*, M. Razeghi, J. S. Lewis, E. Tournié, and G. A. Khodaparast, Eds., International Society for Optics and Photonics, vol. 10926, SPIE, 2019, 109262T. DOI: 10.1117/12.2511301.

[113] C. Scarcella, A. Tosi, F. Villa, S. Tisa, and F. Zappa, “Low-noise low-jitter 32-pixels CMOS single-photon avalanche diodes array for single-photon counting from 300 nm to 900 nm”, *Review of Scientific Instruments*, vol. 84, no. 12, p. 123 112, 2013. DOI: 10.1063/1.4850677.

[114] T. Seets, A. Ingle, M. Laurenzis, and A. Velten, *Motion Adaptive Deblurring with Single-Photon Cameras*, 2020. DOI: 10.48550/ARXIV.2012.07931. [Online]. Available: https://arxiv.org/abs/2012.07931.

[115] S. Ma, S. Gupta, A. C. Ulku, C. Bruschini, E. Charbon, and M. Gupta, “Quanta Burst Photography”, *ACM Trans. Graph.*, vol. 39, no. 4, 2020. DOI: 10.1145/3386569.3392470.

[116] M. Wayne, A. Ulku, A. Ardelean, P. Mos, C. Bruschini, and E. Charbon, “A 500 × 500 Dual-Gate SPAD Imager With 100% Temporal Aperture and 1 ns Minimum Gate Length for FLIM and Phasor Imaging Applications”, *IEEE Transactions on Electron Devices*, vol. 69, no. 6, pp. 2865–2872, 2022. DOI: 10.1109/TED.2022.3168249.

[117] L. Gasparini, M. Zarghami, H. Xu, L. Parmesan, M. M. Garcia, M. Unternährer, B. Bessire, A. Stefanov, D. Stoppa, and M. Perenzoni, “A 32×32-pixel time-resolved single-photon image sensor with 44.64µm pitch and 19.48features reaching 800kHz observation rate for quantum physics applications”, in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 98–100. DOI: 10.1109/ISSCC.2018.8310202.

[118] M. Zarghami, L. Gasparini, and D. Stoppa, “Optimal readout schemes in SPAD-based time-correlated event detection sensor for quantum imaging applications”, in *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, 2017, pp. 373–376. DOI: 10.1109/PRIME.2017.7974185.

[119] S. Lindner, S. Pellegrini, Y. Henrion, B. Rae, M. Wolf, and E. Charbon, “A high-PDE, backside-illuminated SPAD in 65/40-nm 3D IC CMOS pixel with cascoded passive quenching and active recharge”, *IEEE Electron Device Letters*, vol. 38, no. 11, pp. 1547–1550, 2017. DOI: 10.1109/LED.2017.2755989.

[120] M. Hayat, D. Ramirez, G. Rees, and M. Itzler, “Modeling Negative Feedback in Single Photon Avalanche Diodes”, *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7681, Apr. 2010. DOI: 10.1117/12.851914.

[121] Opalkelly.com, *OpalKelly*, 2021. [Online]. Available: https://opalkelly.com/products/xem7360/.

[122] H. Xu, L. Pancheri, G.-F. D. Betta, and D. Stoppa, “Design and characterization of a p+/n-well SPAD array in 150nm CMOS process”, *Opt. Express*, vol. 25, no. 11, pp. 12 765–12 778, 2017. DOI: 10.1364/OE.25.012765.

[123] *PTC2.5K-CH*, 2022. [Online]. Available: https://www.teamwavelength.com/product/ptc2-5k-ch-2-5-a-temperature-controller-chassis-mount/.

[124] flir.eu, *FLIR ETS320*, 2022. [Online]. Available: https://www.flir.eu/products/ets320/.

[125] I. M. Antolovic, C. Bruschini, and E. Charbon, “Dynamic range extension for photon counting arrays”, *Opt. Express*, vol. 26, no. 17, pp. 22 234–22 248, 2018. DOI: 10.1364/OE.26.022234.

[126] *CM100 - Single-channel confocal microscope for reflected-light imaging*, 2022. [Online]. Available: https://www.thorlabs.com/thorproduct.cfm?partnumber=CM100.

[127] *WFA2001 - Epi-illuminator module*, 2022. [Online]. Available: https://www.thorlabs.com/thorproduct.cfm?partnumber=CM100.

[128] *PILAS*, 2022. [Online]. Available: https://www.nktphotonics.com/products/pulsed-diode-lasers/pilas/.

[129] P. W. R. Connolly, X. Ren, A. McCarthy, H. Mai, F. Villa, A. J. Waddie, M. R. Taghizadeh, A. Tosi, F. Zappa, R. K. Henderson, and G. S. Buller, "High concentration factor diffractive microlenses integrated with CMOS single-photon avalanche diode detector arrays for fill-factor improvement", *Appl. Opt.*, vol. 59, no. 14, pp. 4488–4498, 2020. DOI: 10.1364/AO.388993.

[130] C. Zhang, S. Lindner, I. M. Antolović, J. Mata Pavia, M. Wolf, and E. Charbon, "A 30-frames/s, 252 × 144 SPAD Flash LiDAR With 1728 Dual-Clock 48.8-ps TDCs, and Pixel-Wise Integrated Histogramming", *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1137–1151, 2019. DOI: 10.1109/JSSC.2018.2883720.

[131] L. Sbaiz, F. Yang, E. Charbon, S. Susstrunk, and M. Vetterli, "The gigavision camera", in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1093–1096. DOI: 10.1109/ICASSP.2009.4959778.

[132] E. R. Fossum, J. Ma, S. Masoodian, L. Anzagira, and R. Zizza, "The Quanta Image Sensor: Every Photon Counts", *Sensors*, vol. 16, no. 8, 2016, ISSN: 1424-8220. DOI: 10.3390/s16081260. [Online]. Available: https://www.mdpi.com/1424-8220/16/8/1260.

[133] M.-J. Lee, A. R. Ximenes, P. Padmanabhan, T. J. Wang, K. C. Huang, Y. Yamashita, D. N. Yaung, and E. Charbon, "A back-illuminated 3d-stacked single-photon avalanche diode in 45nm cmos technology", in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 16.6.1–16.6.4. DOI: 10.1109/IEDM.2017.8268405.

[134] Y. Kagawa, S. Hida, Y. Kobayashi, K. Takahashi, S. Miyanomae, M. Kawamura, H. Kawashima, H. Yamagishi, T. Hirano, K. Tatani, H. Nakayama, K. Ohno, H. Iwamoto, and S. Kadomura, "The Scaling of Cu-Cu Hybrid Bonding For High Density 3D Chip Stacking", in *2019 Electron Devices Technology and Manufacturing Conference (EDTM)*, 2019, pp. 297–299. DOI: 10.1109/EDTM.2019.8731186.

[135] S.-W. Kim, F. Fodor, N. Heylen, S. Iacovo, J. De Vos, A. Miller, G. Beyer, and E. Beyne, "Novel Cu/SiCN surface topography control for 1 μm pitch hybrid wafer-to-wafer bonding", in *2020 IEEE 70th Electronic Components and Technology Conference (ECTC)*, 2020, pp. 216–222. DOI: 10.1109/ECTC32862.2020.00046.

[136] Y. Kagawa, T. Kamibayashi, Y. Yamano, K. Nishio, A. Sakamoto, T. Yamada, K. Shimizu, T. Hirano, and H. Iwamoto, "Development of face-to-face and face-to-back ultra-fine pitch Cu-Cu hybrid bonding", in *2022 IEEE 72nd Electronic Components and Technology Conference (ECTC)*, 2022, pp. 306–311. DOI: 10.1109/ECTC51906.2022.00057.

[137] S. A. I. Quadri and M. Z. Jahangir, "Design, Implementation and Performance Comparison of Different Branch Predictors on Pipelined-CPU", in *2017 International Conference on Computer, Electrical and Communication Engineering (ICCECE)*, 2017, pp. 1–7. DOI: 10.1109/ICCECE.2017.8526196.

[138] J. Balfour, R. Harting, and W. Dally, "Operand Registers and Explicit Operand Forwarding", *IEEE Computer Architecture Letters*, vol. 8, no. 2, pp. 60–63, 2009. DOI: 10.1109/L-CA.2009.45.

[139] S. Hily and A. Seznec, "Out-of-order execution may not be cost-effective on processors featuring simultaneous multithreading", in *Proceedings Fifth International Symposium on High-Performance Computer Architecture*, 1999, pp. 64–67. DOI: 10.1109/HPCA.1999.744331.

[140] Y. Gao and X. Li, "Formal Verification of Out-of-Order Processor", in *2009 International Conference on Computer Modeling and Simulation*, 2009, pp. 129–135. DOI: 10.1109/ICCMS.2009.47.

[141] T. Selvameena and R. A. Prasath, "Out-of-order execution on reconfigurable heterogeneous MPSOC using particle swarm optimization", in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–6. DOI: 10.1109/ICIIECS.2017.8276021.

[142] J. Farrell and T. Fischer, "Issue logic for a 600-MHz out-of-order execution microprocessor", *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 707–712, 1998. DOI: 10.1109/4.668985.

[143] S. G. Nayak, "Dynamic Branch Prediction for Embedded System Applications", in *2019 International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 966–969. DOI: 10.1109/ICCES45898.2019.9002301.

[144] H. Kim, J. A. Joao, O. Mutlu, C. J. Lee, Y. N. Patt, and R. Cohn, "Virtual Program Counter (VPC) Prediction: Very Low Cost Indirect Branch Prediction Using Conditional Branch Prediction Hardware", *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1153–1170, 2009. DOI: 10.1109/TC.2008.227.

[145] H. Arora, S. Kotecha, and R. Samyal, "Dynamic Branch Prediction Modeller for RISC Architecture", in *2013 International Conference on Machine Intelligence and Research Advancement*, 2013, pp. 397–401. DOI: 10.1109/ICMIRA.2013.84.

[146] P. J. Koopman, Ed., *Stack computers: the new wave.* Pittsburgh: Ellis Horwood, 1989, ISBN: 0745804187.

[147] F. Gramuglia, *High-Performance CMOS SPAD-Based Sensors for Time-of-Flight PET Applications*, 2022. [Online]. Available: https://infoscience.epfl.ch/record/292253?ln=en.

[148] *PULP platform*, 2022. [Online]. Available: https://pulp-platform.org.

[149] *PicoBlaze*, 2022. [Online]. Available: https://www.xilinx.com/products/intellectual-property/picoblaze.html.

[150] *ShivyC*, 2022. [Online]. Available: https://github.com/ShivamSarodia/ShivyC.

[151] F. Gutierrez-Barragan, A. Ingle, T. Seets, M. Gupta, and A. Velten, "Compressive single-photon 3d cameras", in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 833–17 843. DOI: 10.1109/CVPR52688.2022.01733.

[152] D. Stoppa, *A reconfigurable QVGA/Q3VGA direct Time-of-Flight 3D imaging system with on-chip depth-map computation in 45/40nm 3D-stacked BSI SPAD CMOS*, 2021.

[153] P.-Y. Taloud, S. Bernhard, A. Biber, M. Boehm, P. Chelvam, A. Cruz, A. D. Chele, R. Gancarz, K. Ishizaki, P. Jantscher, T. Jessenig, R. Kappel, L. Lin, S. Lindner, H. Mahmoudi, A. Makkaoui, J. Miguel, P. Padmanabhan, L. Perruchoud, D. Perenzoni, G. Roehrer, A. Srowig, B. Vaello, and D. Stoppa, *A 1.2K dots direct Time-of-Flight 3D Imaging System with on-chip depth map computation, in 45/22nm 3D-stacked BSI SPAD CMOS*, 2022.

[154] C. Peters, J. Klein, M. B. Hullin, and R. Klein, "Solving Trigonometric Moment Problems for Fast Transient Imaging", *ACM Trans. Graph.*, vol. 34, no. 6, 2015. DOI: 10.1145/2816795.2818103.

[155] A. Simonetto, G. Agresti, P. Zanuttigh, and H. Schäfer, "Lightweight Deep Learning Architecture for MPI Correction and Transient Reconstruction", *IEEE Transactions on Computational Imaging*, vol. 8, pp. 721–732, 2022. DOI: 10.1109/TCI.2022.3197928.

[156] Z. Wu, C. Zuo, W. Guo, T. Tao, and Q. Zhang, "High-speed three-dimensional shape measurement based on cyclic complementary Gray-code light", *Opt. Express*, vol. 27, no. 2, pp. 1283–1297, 2019. DOI: 10.1364/OE.27.001283.

[157] Q. Zhang, X. Su, L. Xiang, and X. Sun, "3-D shape measurement based on complementary Gray-code light", *Optics and Lasers in Engineering*, vol. 50, no. 4, pp. 574–579, 2012, Computational Optical Measurement, ISSN: 0143-8166. DOI: https://doi.org/10.1016/j.optlaseng.2011.06.024.

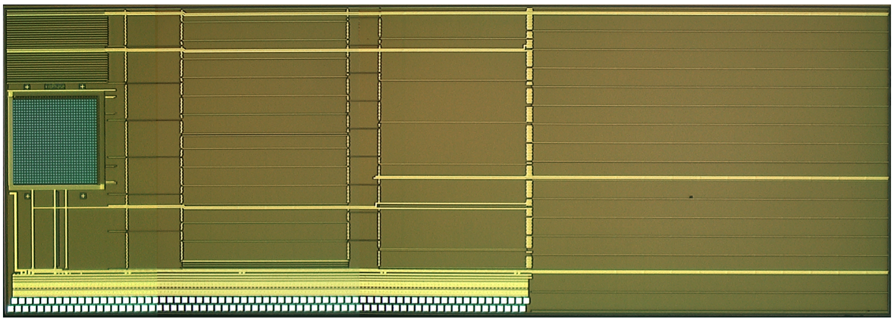
[158] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[159] *VisIR-780*, 2022. [Online]. Available: https://www.picoquant.com/products/category/high-power-and-uv-lasers/visir-versatile-picosecond-laser-module#custom1.

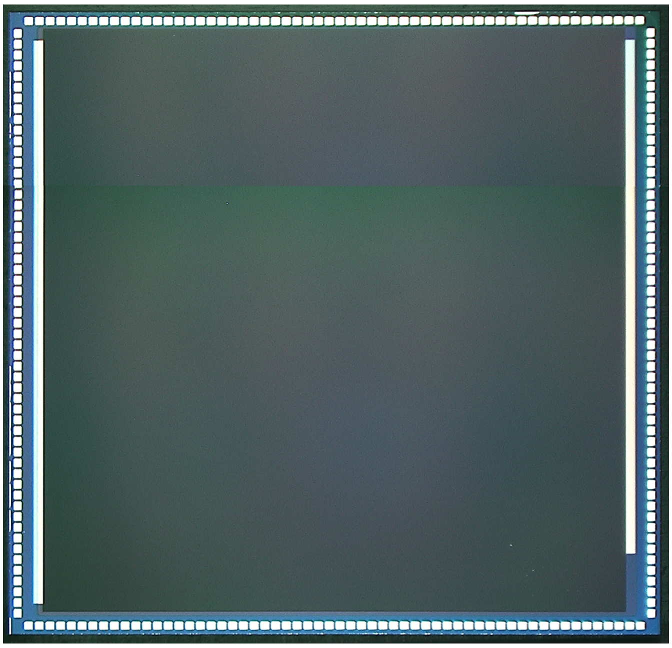
[160] F. Gramuglia, P. Keshavarzian, E. Kizilkan, C. Bruschini, S. S. Tan, M. Tng, E. Quek, M.-J. Lee, and E. Charbon, “Engineering Breakdown Probability Profile for PDP and DCR Optimization in a SPAD Fabricated in a Standard 55 nm BCD Process”, *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 28, no. 2: Optical Detectors, pp. 1–10, 2022. DOI: 10.1109/JSTQE.2021.3114346.

[161] K. Morimoto, M.-L. Wu, A. Ardelean, and E. Charbon, “Superluminal Motion-Assisted Four-Dimensional Light-in-Flight Imaging”, *Phys. Rev. X*, vol. 11, p. 011 005, 1 2021. DOI: 10.1103/PhysRevX.11.011005.

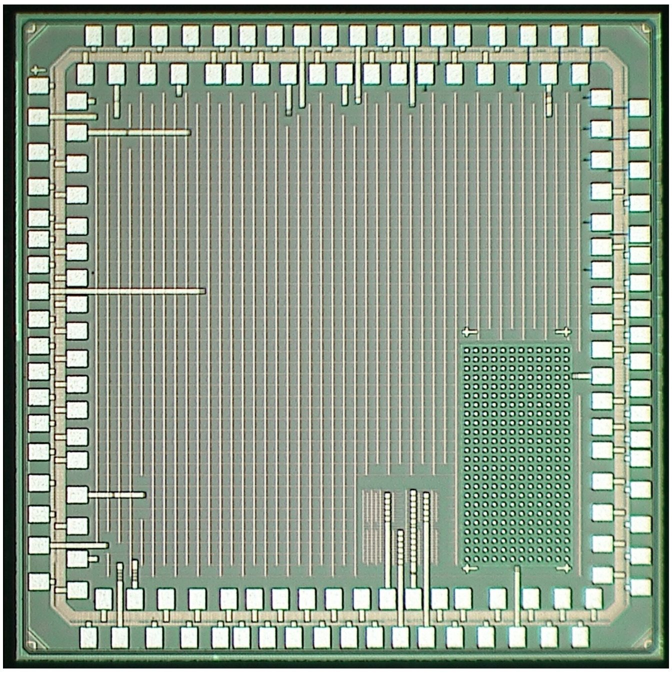
Chip gallery



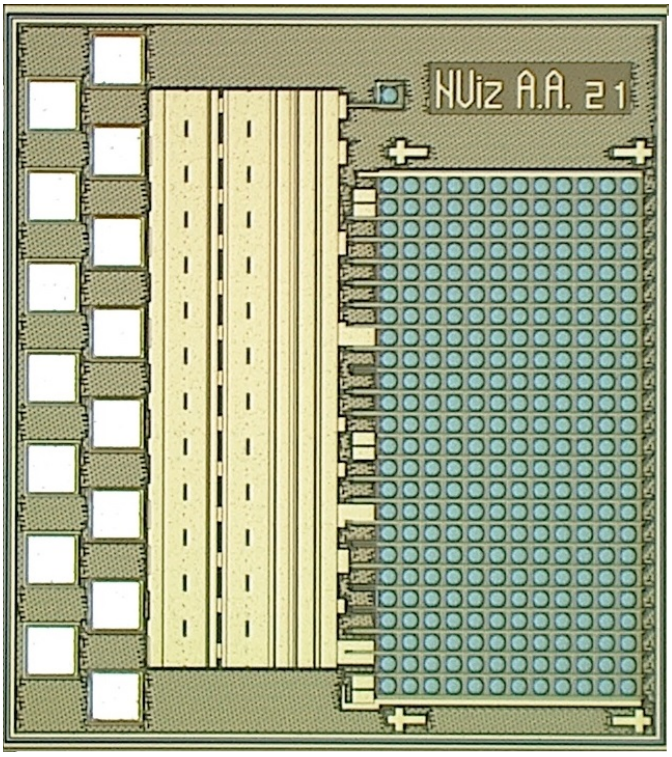
*kilo*Phase (Chapter 2). Die size is 9.5 × 3.3 mm^2.



*Mega*Phase (Chapter 3). Die size is 8 × 7.5 mm^2.



*Ultra*Phase bottom tier (Chapter 4). Die size is 2 × 2 mm^2.



*Ultra*Phase top tier (Chapter 4). Die size is 0.86 × 0.95 mm^2.

List of publications

Journal articles as first author

- A. C. Ulku*, **A. Ardelean***, M. Antolovic, S. Weiss, E. Charbon, C. Bruschini and X. Michalet, "Wide-Field Time-Gated SPAD Imager for Phasor-Based FLIM Applications", *Methods and Applications in Fluorescence*, vol. 8, no. 2, p. 024002, 2020. (*Equally contributing authors)
- K. Morimoto*, M. L. Wu*, **A. Ardelean*** and E. Charbon, "Superluminal Motion-Assisted Four-Dimensional Light-in-Flight Imaging", *Physical Review X*, vol. 11, 2021. (*Equally contributing authors)

Journal articles as co-author

- K. Morimoto, **A. Ardelean**, M. L. Wu, A. C. Ulku, M. Antolovic, C. Bruschini and E. Charbon, "Megapixel Time-Gated SPAD Image Sensor for 2D and 3D Imaging Applications", *Optica*, vol. 7, 2020.
- A. Muntean, E. Venialgo, **A. Ardelean**, A. Sachdeva, E. Ripiccini, D. Palubiak, C. Jackson and E. Charbon, "Blumino: The First Fully Integrated Analog SiPM With On-Chip Time Conversion", *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. PP, p. 1-1, 2020.
- M. Wayne, A. C. Ulku, **A. Ardelean**, P. Mos, C. Bruschini and E. Charbon, "A 500 × 500 Dual-Gate SPAD Imager With 100% Temporal Aperture and 1 ns Minimum Gate Length for FLIM and Phasor Imaging Applications", *IEEE Transactions on Electron Devices*, vol. 69, p. 2865-2872, 2022.
- M. Wayne,A. C. Ulku, P. Mos, **A. Ardelean**, E. Sie, C. Bruschini, F. Marsili and E. Charbon, "High-sensitivity multispeckle diffuse correlation spectroscopy with a 500 × 500 SPAD array", submitted to *Biomedical Optics Express*, 2022.

Conference articles as first author

- **A. Ardelean**, A. C. Ulku, X. Michalet, E. Charbon and C. Bruschini, "Fluorescence Lifetime Imaging with a Single-Photon SPAD Array Using Long Overlapping Gates: an Experimental and Theoretical Study", *Proceedings of SPIE–the International Society for Optical Engineering*, vol. 10882, p. 33, 2019.

Conference articles as co-author

- F. Gramuglia, A. Muntean, E. Venialgo, M.J. Lee, S. Lindner, M. Motoyoshi, **A. Ardelean**, C. Bruschini and E. Charbon, "CMOS 3D-Stacked FSI Multi-Channel Digital SiPM for Time-of-Flight PET Applications", *Proceedings of IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, 2020.

Conference presentations

- A. C. Ulku*, **A. Ardelean***, P. Mos, E. Charbon and C. Bruschini, "SwissSPAD3 - a Dual-Gate Photon-Counting SPAD Sensor for Widefield FLIM Imaging", *Focus on Microscopy*, 2021, Conference presentation. (*Equally contributing authors)
- A. C. Ulku, **A. Ardelean**, M. Antolovic, S. Weiss, E. Charbon, C. Bruschini and X. Michalet, "Wide-Field Time-Gated SPAD Imager for Phasor-Based FLIM Applications", *Biophysical Society Annual Meeting*, 2020, Poster.

Curriculum vitae

Andrei Ardelean

1992 Born in Timișoara, Romania

Education

2017 - 2022 **Ecole Polytechnique Fédérale de Lausanne (EPFL)**
Advanced Quantum Architecture Laboratory (AQUA)
Lausanne, Switzerland
Ph.D. in Microelectronics
Thesis: "*Computational Imaging SPAD Cameras*"

2015 - 2017 **Delft University of Technology (TUDelft)**
Circuit and Systems Group (CAS)
Delft, The Netherlands
M.Sc. in Microelectronics
Thesis: "*Energy-efficient Multipath Ring Network for Heterogeneous Clustered Neuronal Arrays*"

2011 - 2015 **Politehnica University of Timișoara (UPT)**
Timișoara, Romania
B.Sc in Electronics Engineering
Thesis: "*Control of a Ball-and-Plate System with an Artificial Neural Network on FPGA*"

Professional Experience

2016 - 2016	**NASA Jet Propulsion Laboratory** Pasadena, California, USA Advanced UV/Vis/NIR Detector Arrays, Systems and Nanoscience Group Intern
2014 - 2015	**HELLA Romania** Timișoara, Romania Advanced Engineering Group Hardware Developer