Bio-inspired Electronics For Micropower Vision Processing

by

Timothy Constandinou

October 2005

A thesis submitted for
the degree of Doctor of Philosophy of the University of London

Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
University of London

Contents

List of Figures

List of Tables

To my parents and Marianna!

Acknowledgements

First and foremost, I would like to acknowledge my supervisor Chris Toumazou. He has always been a constant source of inspiration, support and encouragement. Chris has taught me that imagination, intuition and common sense are more important than formal education. He has always motivated me to look at the bigger picture whilst guiding me to remain focused.

I would also like to thank Julius Georgiou; a friend, colleague and brilliant engineer. It was Julio who first proposed the idea of doing a PhD to me. Over the years we have worked closely together and I have been fortunate to have his feedback and insight time and time again.

From the University of Oslo (UIO), I want to thank Tor Sverre Lande (Bassen) who introduced me to the world of neuromorphic electronics. His constant enthusiasm in non-conventional, elegant solutions and strong belief in my work had encouraged me to pursue research in this field. Also from UIO, I wish to acknowledge Philipp Häfliger, for providing me complete access to his address event libraries.

I want to thank from the Physics & Bioengineering departments Patrick Degenaar and Dylan Banks for many invaluable discussions and for providing me unrestricted access to taking optoelectronic measurements in the Blackett Laboratory. Patrick initiated the idea of the ON/OFF spiking photoreceptor and I have enjoyed developing it with him.

From the Circuits & Systems group, I would like to thank both past and present members; providing a friendly yet intellectually rich environment. I thank Christos Papavassiliou, Solon Despotopoulos, Ganesh Kathiresan, Emmanuel Drakakis, Kostis Michelakis, Kritsapon (Chy) Leelavattananon, Tony Vilches, Rohit Arora, Okundu Omeni and Mohammed Semati for introducing and welcoming me to the group, David Yates, Calvin Sim and Wim Melis for enduring the past three years alongside me and Michalis Frangos,

Francesco Cannillo, Leila Shepherd, Pantelakis Georgiou, Andreas Katsiamis, Sofia Vatti, Themistoklis Prodromakis, Chun Lee, Aleksandra Rankov, Phil Corbishley, Xu Min and the list goes on.. for continuing to provide a wonderful work environment. Furthermore, I wish to thank the group administrators, Wiesia Hsissen and Angela Bishop for their constant support and warmth they bring to the group. From outside the group, I thank Alex Charalambides and Samia Antoun for taking countless coffee breaks with me when work got too much.

I have been very fortunate to receive both financial support and access to technical resources from the team at Toumaz Technology Ltd, in particular I want to thank Keith Errey and Alison Burdett.

Last and most importantly, I would like to express my gratitude to my family. I thank my parents Gregory and Niki, my sister Marianna and my grandparents Timothy and Yianoulla for their continuing support and encouragement. I finally thank my partner Katy for enduring the past three years and providing me with constant love, support and excitement.

Abstract

Vision processing is a topic traditionally associated with neurobiology; known to encode, process and interpret visual data most effectively. For example, the human retina; an exquisite sheet of neurobiological wetware, is amongst the most powerful and efficient vision processors known to mankind. With improving integrated technologies, this has generated considerable research interest in the microelectronics community in a quest to develop effective, efficient and robust vision processing hardware with real-time capability.

This thesis describes the design of a novel biologically-inspired hybrid analogue/digital vision chip ORASIS[1] for centroiding, sizing and counting of enclosed objects. This chip is the first two-dimensional silicon retina capable of centroiding and sizing multiple objects[2] in true parallel fashion. Based on a novel distributed architecture, this system achieves ultra-fast and ultra-low power operation in comparison to conventional techniques.

Although specifically applied to centroid detection, the generalised architecture in fact presents a new biologically-inspired processing paradigm entitled: *distributed asynchronous mixed-signal logic processing.* This is applicable to vision and sensory processing applications in general that require processing of large numbers of parallel inputs, normally presenting a computational bottleneck.

Apart from the distributed architecture, the specific centroiding algorithm and vision chip other original contributions include: an ultra-low power tunable edge-detection circuit, an adjustable threshold local/global smoothing network and an ON/OFF-adaptive spiking photoreceptor circuit.

Finally, a concise yet comprehensive overview of photodiode design methodology is provided for standard CMOS technologies. This aims to form a basic reference from an engineering perspective, bridging together theory with measured results. Furthermore, an approximate photodiode expression is presented, aiming to provide vision chip designers with a basic tool for pre-fabrication calculations.

[1] ORASIS is taken from the Greek word: *όρασση* (orasi) meaning vision.

[2] Such as biological cells under a microscope, pharmaceutical drugs under production line inspection, etc.

Chapter 1

Introduction

1.1 Motivation Neurobiology

Biology makes excellent use of resources to solve a given task. For example, the human retina; an exquisite sheet of neural tissue, is made up of a layered network of millions of poorly replicated yet statistically identical primitives to provide the brain with a well-conditioned neural image of what we see. Extending beyond the retina into the brain, this employs much the same strategy; using billions of poorly defined primitives to achieve the most computationally demanding and perceptive tasks. Face recognition, real-time navigation control, object segmentation, depth perception, saliency detection are a few tasks that we routinely perform and take for granted; yet even our most advanced computational hardware is incapable of achieving acceptable let alone comparable results. Biology is efficient, robust, adaptable, real-time, effective, scalable and reliable.

For the above reasons, any engineer would do extremely well in learning from nature. In electronics, developing systems based on neurobiological hierarchy, organisation [1], representation and/or structure could result in improved efficiency, robustness, adaptability, responsivity, effectiveness and reliability.

1.2 Research Objectives

This research is aimed in exploring biologically-inspired electronics [2] through developing a specific application, employing a hybrid strategy. This aims to combine biological-inspired

methods together with conventional techniques to achieve the best of both worlds. Subsequently, this work is targeted towards providing some insight into answering the following questions:

- In which aspects is neurobiology superior to modern microelectronic technologies and vice-versa ?
- What can we learn from biology that is efficiently implementable in standard microelectronic technologies and can provide substantial advantage over conventional techniques ?

1.3 Overview

This section aims to provide a concise and brief, single paragraph introduction to the material covered in each of the following chapters.

1.3.1 Biologically Inspired Electronics

This chapter is aimed at introducing the term "biologically inspired electronics" and identifying appropriate technologies for implementing such systems. Initially fundamentals of neurobiology such as structure, organisation and hierarchy are described, leading to the notion of biologically inspired representation. This approach is then extrapolated to microelectronics; in particular emphasising on process efficient representation. The notion of hybrid computation is discussed and the relative merits and flaws concerning resource efficient computation are compared for common operations. Finally for modes of device operation in current microelectronic technologies and associated design techniques related to reliable and efficient implementation in this representation space are discussed and reviewed.

1.3.2 Modern Vision Processing Technology

This chapter begins by comparing modern imaging and vision acquisition process technologies. Following is an extension to a typical processing platform identifying key features and limitations. Distributed techniques resembling biologically-inspired systems are then introduced and outlines as to provide a plausible solution to the limitations of modern systems. Finally an extensive vision chip review is presented for all centroid processing systems

reported to date; employing both traditional computational techniques and embedded in distributed hardware.

1.3.3 A Distributed Architecture for Centroid Detection

This chapter presents a novel distributed algorithm for centroiding and sizing of regular objects. Through a true parallel processing approach, tradition information flow bottlenecks have been overcome and high computational efficiency is achievable. Furthermore, the presented scheme; based on a biologically-inspired organisation demonstrates great robustness to fabrication non-idealities and ill-conditioned sensor data. This has been experimentally verified by testing both to non-uniform input data and pixel processing element mismatch. Finally the presented architecture is extended to form a generic distributed array processing paradigm. The basic concept is described and related to a biological vision system with some example implementable algorithms being outlined.

1.3.4 Photodiodes in Modern Deep Sub-Micron CMOS Technology

During the past few years, an increased interest in CMOS technology for imaging applications; principally due to the progress of the Active Pixel Sensor (APS) approach, has prompted much work on photodiode modelling. Although much work has been published on modelling phototransduction, reliable modelling in standard CMOS technologies has not been widely accessible. This is partly due to corporate interests, and partly due to established semiconductor physics being applicable to CMOS phototransduction. This chapter begins by presenting a unified phototransduction theory specifically for PN junction photodiodes within standard CMOS technologies. A series of test devices are then fabricated in a modern technology and all the measured results are presented. These are used to validate the developed model and outline a set of generic design rules for good photodiode design within a standard CMOS technology. Finally, a biologically-inspired photoreceptor circuit is presented; using an ON/OFF spiking scheme to provide adaptable, tradable dynamic range, spatial and temporal resolution. This circuit is designed to be part of a dynamically reconfigurable foveating silicon retina, details of which are not included.

1.3.5 ORASIS: A Micropower Centroiding Vision Processor

The final chapter presents the complete system implementing the distributed algorithm (described in Chapter 4) and photosensing elements (described in Chapter 5) into custom integrated hardware. This system, named ORASIS; presents the first distributed vision processor (or silicon retina) capable of multiple centroid detection and sizing; implemented in a standard CMOS technology. This chapter describes the architecture, hierarchy, interconnectivity and specific circuit blocks also including simulated and measured results.

References

[1] G. M. Shepherd (ed.), *The Synaptic Organization of the Brain.* Oxford University Press, 1998.

[2] C. A. Mead, *Analog VLSI and Neural Systems.* Addison-Wesley, 1989.

Chapter 2

Biologically Inspired Electronics

2.1 Introduction

Although modern microelectronic technologies have surpassed our expectations in virtually all areas, there still remains a vast application space of computational problems either too challenging or complex to be solved with conventional means. These applications often require the transformation of data across the boundary between the real (analogue) world and the digital world. The problem arises, whenever a system is sampling and acting on real-world data, for example in any recognition or identification task. Traditional processing techniques find it very challenging and computationally demanding to identify and process complex structures and relationships in vast quantities of ill-conditioned data (low precision, ambiguous and noisy) [1].

Although great progress has been made in hardware processing techniques (eg. DSP, FPGA), in both computational power and efficiency, the solution to complex recognition tasks still continues to elude us. Furthermore, neither artificial intelligence, artificial neural networks nor fuzzy logic has provided us with an effective and robust solution. However, biological organisms routinely accomplish complex visual tasks such as object recognition and target tracking. For example, a common housefly; with a brain the size of a grain of rice can outperform our modern multiple gigahertz processors in real-time obstacle avoidance in flight navigation in addition to countless other perception tasks.

Thus, the Neuromorphic community has emerged aiming to provide a design methodology for tackling such problems. Using hybrid, distributed processing architectures based on simple primitives; inspired by biology, modern microelectronic technology is progressing

Computation	Neural	Electronic (von Neumann based)	Ref.
Representation	Analogue, spike domain[1]	Digital	[3]
Processing	Parallel	Serial	[3]
Power	Low	High	[4]
Speed	Low, High[2]	High	[4]
Learning	Adaptive	Preset	[5]
Precision	Fuzzy	Accurate	[5]
Sensitivity	Redundant	Fault sensitive	[6]
Organisation	Monolithic	Modular	[7]
Connectivity	Very high fan-out	Sparsely connected	[7]
Geometry	3 dimensions[3]	2 dimensions	[6]

[1] Intracellular representation is continuous whereas intercellular is discrete.
[2] Dependant on the particular process and level of abstraction, eg. low speed operation at neuronal level, however extremely high speed at cortical level.
[3] Often classed as 2.5 dimension as neural tissue typically has layered structure that folds, occupying a 3 dimensional space.

Table 2.1: Comparison of Neural and Electronic Computational Paradigms

one step closer to finding a workable solution to these problems.

2.2 Neural Organisation [2]

Neurobiology has a massively different organisation to that of any conventional electronic system. This vast difference is highly evident when comparing various features of the two systems (see Table 2.1).

This section aims to provide a basic overview of the neurobiological organisation, starting from the fundamental processing element and building block; the biological neuron. The mechanism of synaptic processes; providing intra- and inter-neuronal communication and

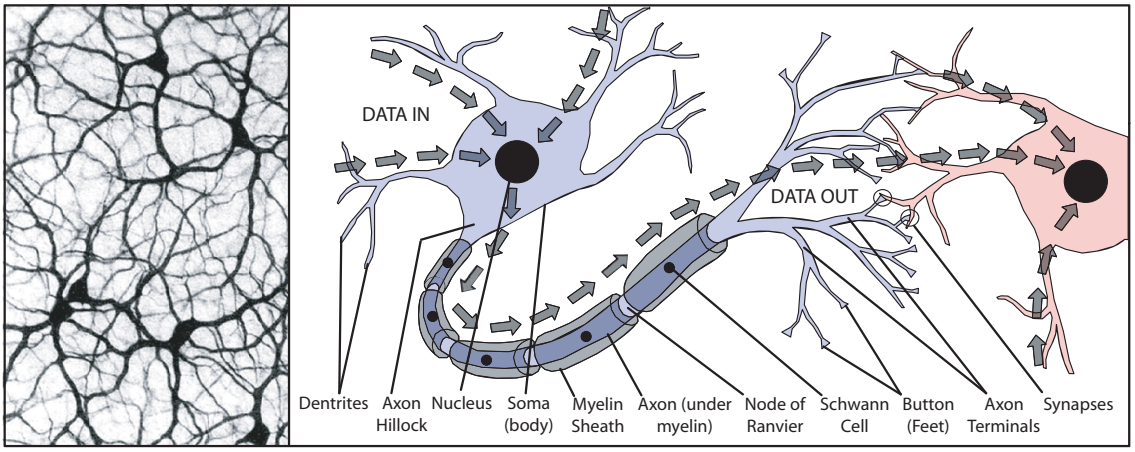


Figure 2.1: Neural architecture - Horizontal cells from a rabbits retina representing the intricate web-like neural interconnectivity (left) and typical representation of a neuron highlighting data flow (right)

interaction will then be discussed. Having established these key primitives, the neural hierarchical organisation will be outlined; linking together genes, molecules, synapses, neurons, neural networks and beyond.

2.2.1 Neurons [8]

The brain is a collection of about 10 billion interconnected neurons. A neuron is a cell (see Fig. 2.1) which uses biochemical reactions to receive, process and transmit information.

Each neuron's dendritic tree is connected to up to thousands of neighbouring neurons. When any of these neurons fire, a positive or negative charge is received by one of the dendrites. The strengths of all the received charges are added together through the processes of spatial and temporal summation. The aggregate input is then passed to the soma (cell body). The soma and the enclosed nucleus don't play a significant role in the processing of incoming and outgoing data. Their primary function is to perform the continuous maintenance required to keep the neuron functional. The part of the soma that does concern itself with the signal is the axon hillock (and the internal surface membrane.) If the aggregate input is greater than the axon hillock's threshold value, then the neuron "fires" and an output signal (known as the action potential) is transmitted down the axon. The strength of the output is constant, regardless of whether the input was just above the threshold, or say a hundred times as great, however the time to spike is affected. The output strength is

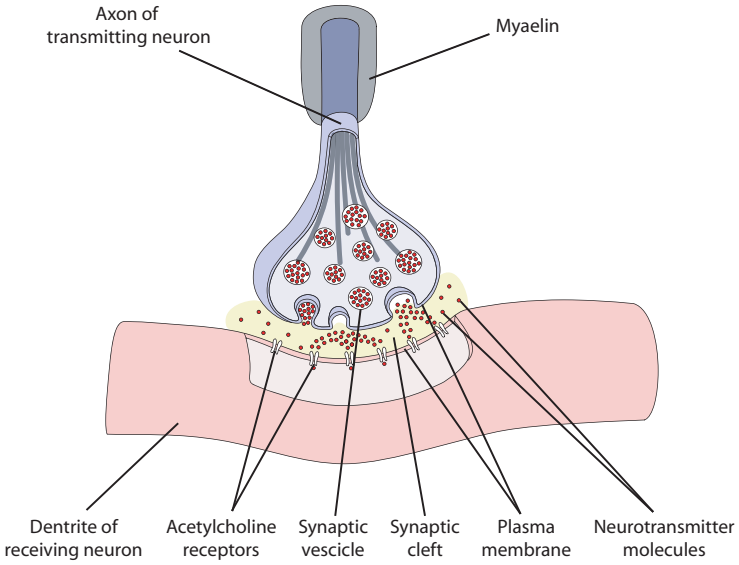


Figure 2.2: The synapse, a neurobiological mechanism forming the *contact* sites that facilitate interneuronal connections for transmission and processing of neural information.

unaffected by the many divisions in the axon; it reaches each terminal button with the same intensity it had at the axon hillock. This uniformity is critical in an analogue device such as a brain where small errors can snowball, and where error correction is more difficult than in a digital system. It is this action of the neuron firing when a threshold value is reached which introduces the gain that enables the processing in the neurobiological system.

2.2.2 Synapses [7]

Each terminal button is connected to other neurons across a small gap called a synapse (see Fig. 2.2) The physical and neurobiological properties of each synapse, determines the strength and polarity of the new input signal. This is where the nervous system is the most flexible. Changing the constitution of various neurotransmitter chemicals can increase or decrease the amount of stimulation that the firing axon imparts on the neighbouring dendrite. Altering the neurotransmitters can also change whether the stimulation is excitatory or inhibitory. It is this dynamic nature of the synapses that provides the neurobiological architecture with the ability to adapt and "learn". This results in an extremely robust system architecture, being considerably immune to component degradation and failure.

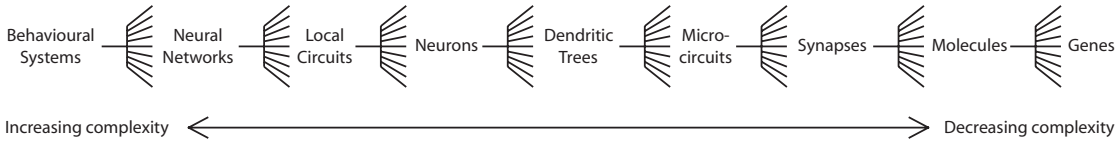


Figure 2.3: The hierarchical organisation of the nervous system, from the highest level; behavioural systems (on the left,) to the lowest level; genes (on the right)

2.2.3 Neural Hierarchy [7]

Neurobiology uses an intricate *hierarchical* organisation, with spatial and temporal scales spanning several orders of magnitude to produce a certain behaviour in a given organism. This begins at the genetic level, with *genes* interacting with the environment to define the basic protein constitution in the different regions of the nervous system. These *molecular* components form the various parts of the different brain cells and furthermore, the synaptic organisation is facilitated through the genetic blueprint. The smallest clusters of interconnected *synapses* then form local units often referred to as *microcircuits* [9]. These are grouped to form *dendritic subunits* [10] within a dendritic trees of individual neurons. A single *neuron* may contain several such dendritic subunits. The next level in the hierarchy consists of *local circuits* [11], being groups of interconnected neurons of similar types. Such neuronal groups are then arranged into larger neuronal networks including *interregional pathways, columns, laminae* and *topographic maps* involving multiple regions in the brain that mediate specific types of behaviour. A basic representation of this intricate hierarchy is illustrated in Fig. 2.3.

2.3 Neural Representation

To provide an insight into neurobiological data representation, this section begins by discussing how visual information is segregated and transmitted in parallel visual pathways. This is then extended to the cellular level; in particular, with reference to the retina- the underlying inspiration being the *perfect* use of resources.

This philosophy is then applied to microelectronics. Much work has already gone into trying to realise neurobiological systems in silicon technologies [12] [13]; however the two-dimensional geometry with limited interconnection capacity has proved no match for three-dimensional neurobiological wetware [4]. Subsequently a different approach is taken; to aim

to use the general organisation and representation in biology to help optimise our designs to our available resources.

2.3.1 Neural Visual Streams [2]

Anatomical studies show that neurons in the visual pathway (see Fig. 2.4) are segregated into several visual streams. The functional role of the visual streams must be inferred from the anatomical properties along with the way neurons in these separate streams respond to light stimulation.

Different visual streams each have a unique role; encoding specific features extracted from the image, then being relayed to different parts of the brain and central nervous system.

The most important information represented by visual pathways is image contrast rather than absolute light level. Visual contrast is the ratio of localised light level to average image intensity. To represent the image contrast, neurons in the visual pathway change sensitivity to compensate for changes in the mean illumination level. This process, called visual adaptation, allows the biological visual system to represent scenes of extremely high total dynamic range without compromising fine details.

Contrast is supplied via two complimentary visual streams up to the primary visual cortex. One of these represents contrast information varying slowly over space but rapidly over time, whilst the other varies rapidly over space but slowly over time. These can each have their own purpose; for example, one stream can relay fine detail for object recognition tasks, whilst the other provides transient information for saliency and attention processing.

Beyond the primary visual cortex, behavioural and electrophysiological measurements suggest that image contrast is represented within separate visual streams that each specialise in coding the information within a certain range of spacial frequencies and orientations. The multi-resolution representations is qualitatively consistent with measurements of receptive-field properties in the primary visual cortex. Multi-resolution image representations have become a standard tool in computational applications, including image compression, segmentation and analysis.

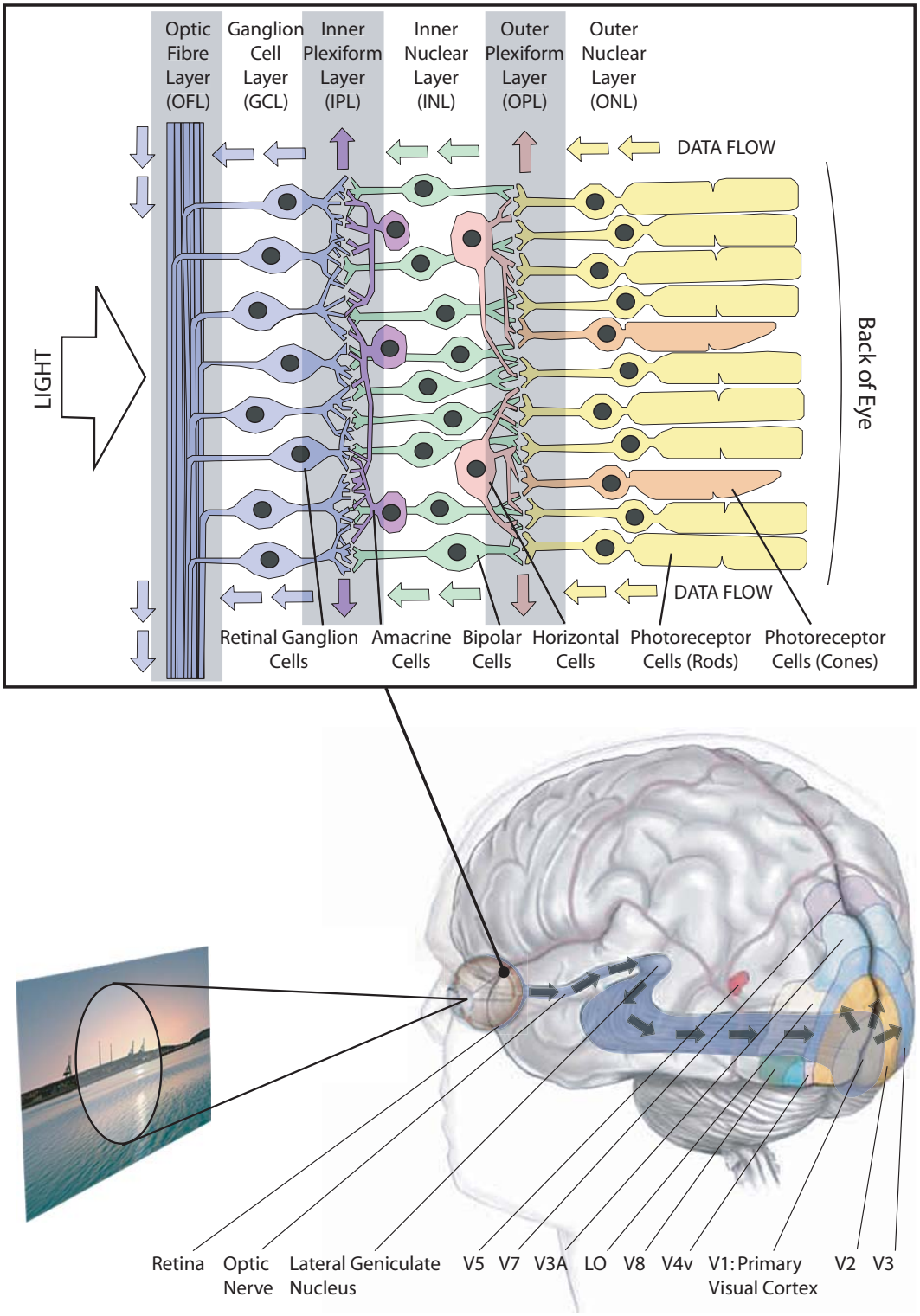


Figure 2.4: The human visual pathway - cellular representation of the retina (top) and various parts of the brain associated with visual processing (bottom)

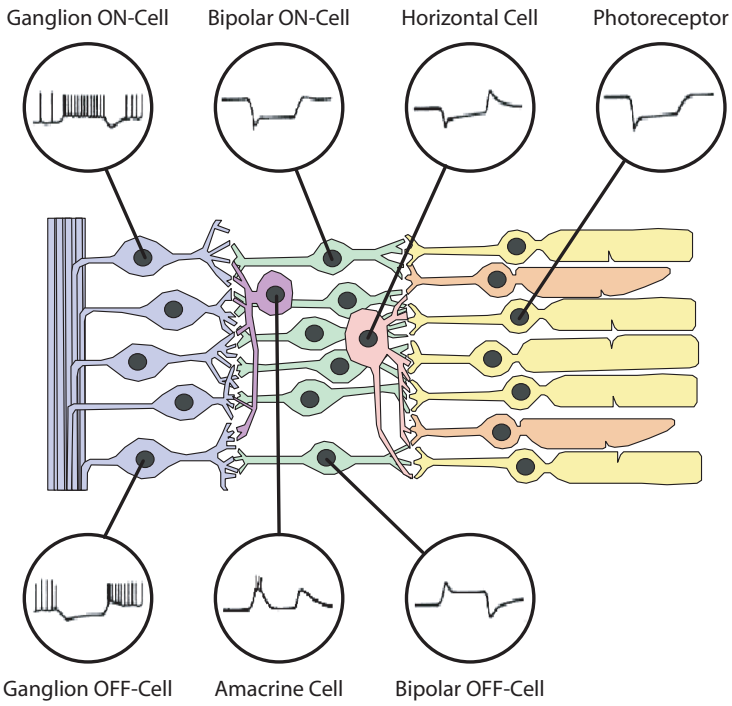


Figure 2.5: Signal representation in the primate retina, illustrating typical electrical response of various neuron types to spot illumination.

2.3.2 Retinal Data Representation

A common paradigm is the associating of neural activity with spiking and action potentials. Although fundamentally correct, it is commonly overlooked that a great deal of neural processing in fact is continuous in both time and value. Here, it is not the spike timing which encode the data but rather the synaptic biochemistry at the various neural interfaces.

A prime example is the mammalian retina; consisting of over 75 discrete neuron types classed into five main groups. Of these, the only group to transmit data as action potentials are the ganglion cells. The other neuron groups (photoreceptors, horizontal, bipolar and amacrine cells) encode, process and share data as graded potentials (see Fig. 2.5) Transduction proteins and ion channels are optimised for sensitivity, speed, gain and noise. At each interface, the different representations serve to achieve optimum performance for a given function. It is this remarkable organisation that ensures both exceptional performance and great computational efficiency. A good review on retinal circuit optimisation can be found in reference [14].

2.3.3 Quantised Data/time vs. Continuous Data/time

Information exists in a three dimensional media with data encoded in time, intensity and space. Various techniques of data representation utilise this space in different ways. Since spatial content can be included in all electronic representations; i.e. it is not fundamental, it will only be included in the next subsection on spike-domain coding.

For example, in electronics, analogue circuits represent data as continuous voltages and currents, varying both in time and intensity. On the other hand, conventional digital electronics use clocks to synchronise activity; data therefore being represented as discrete voltages; being quantised in time in addition to value.

Sampled data techniques exist such as switched-capacitor (SC) [15] and switched-current (SI) [16] that use a clock to sample continuous-varying signals and are therefore discrete in time but continuous in amplitude. Such techniques are widely used in signal processing of continuous (analogue) signals, for example in implementing filters for oversampling data converters.

Exploring this two dimensional space (time and amplitude, see Fig. 2.6) for encoding of data; the only remaining unexploited representation is continuous-time, discrete-data. This is in fact the principle representation of biology; with spiking neurons conveying no data in the shape or amplitude of the action potential, but rather in the timing. This encoding can easily be achieved using asynchronous digital technology; although not widely used in system-level design due to complexity in synthesis.

2.3.4 Spike Domain

This wide category of this continuous-time spike coding is often referred to as spike domain. This has the property that all processing is in fact event driven, i.e. the data directly triggers the processing rather than using some external stimuli or signal as is the case in the majority of integrated systems.

Spike domain coding is often only associated with the frequency of spike occurrence, referred to as *rate coding*. However, this alone cannot account for the high temporal performance achieved in neurobiology, referred to as temporal hyper-acuity. For example, echo-locating bats have been reported [17] to be able to discriminate echo delays from 10 to 50 nanoseconds. If the minimum spike rate is in the order of milliseconds, some coding

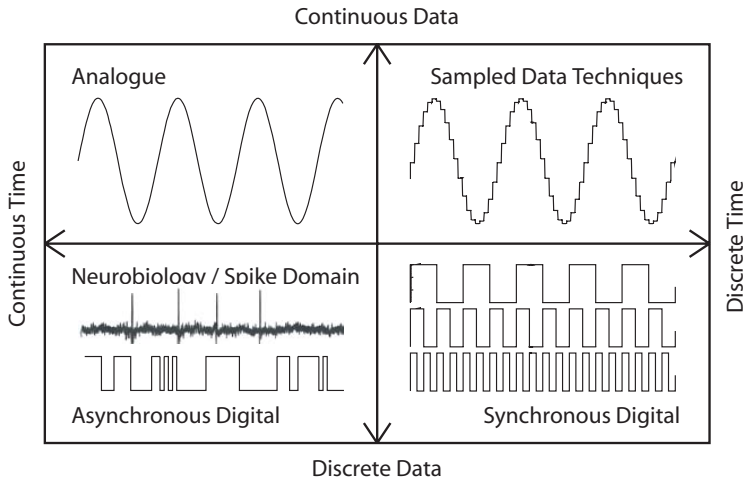


Figure 2.6: Classification of the various data representation techniques in standard microelectronic technologies

scheme other than rate coding must be responsible for this phenomenon.

Several different schemes [18] have been proposed utilising both temporal and spatial properties, in order to interpret how data is actually encoded in biology; the most popular being listed below:

- *Rate coding (temporal):* The most popular idea is that the data is represented by the density (or rate) of spikes (or pulses.) Experiments that measured the response of V1 cells in the Primary Visual Cortex [19] [20], found that increasing the contrast input at the eye had the effect of increasing the rate at which the neurons fired. By periodically counting the number of spikes in a set sampling window, the data is decoded. This is good for slowly changing stimuli, or where long distance transmission is required as this technique inherently removes random noise.

- *Population average coding (spatial):* In this scheme, the data is encoded as a normalised spatial average. By periodically counting the number of neurons firing within a short window and normalising to the population size, the population average is determined. This scheme has good temporal properties and is therefore sensitive to rapid changes.

- *Time of arrival (spatiotemporal):* This coding normally conveys data concerning an external event, with most of the information contained in the first few spikes.

- *Phase coding (spatiotemporal):* In this scheme, the data is encoded as the phase difference between different spike trains (or pulse streams).
- *Synchrony (spatiotemporal):* In this scheme the data is encoded in a pattern of spikes produced by a set of neurons, for example, the pattern may be for simultaneous or correlated firing.

2.4 Hybrid Computation for Improved Computational Efficiency

Having illustrated how biology uses a sparse set of signal representation forms at different levels; it follows that the same strategy should be applied in microelectronics. This section starts by comparing different signal representation and processing modalities in standard CMOS technologies. Following is a qualitative comparison of selected linear and non-linear mathematical operations implemented in different signal and data representation techniques. Finally, various methods of combining these techniques in a realistic system level organisation will be outlined.

2.4.1 Analogue versus Digital Signal Processing

A key debate in the low power electronics community is whether analogue or digital signal processing can be more computationally efficient.

Much work [21] [1] [22] [23] [4] has already gone into this by considering factors such as signal-to-noise ratio (SNR,) power consumption, silicon area, channel utilisation, design time, etc. Following from this, the general conclusions are:

- *Analogue processing* can be far more computationally efficient than digital signal processing. This is due to the rich mathematical content in the physics of the devices in comparison to the primitive nature of a digital device (a switch). It follows that to achieve similar functionality with digital logic, many more devices need to be used; in fact this can be several orders of magnitude more devices! Moreover, at high activities this results in significantly higher power consumption. This is because digital logic dissipates both due to continuous subthreshold "leakage" current (static power) and during switching (dynamic power,) whereas analogue devices only have a

continuous current supply (static power.)

- *Digital processing* is immune to noise and cumulative offsets. The continuous nature of analogue signals means they cannot be restored at each stage as discrete signals can. Consequently any noise or circuit-introduced offset accumulates through cascading and can ultimately deteriorate the signal in complex analogue systems. This reduces the accuracy and dynamic range of such a system for a given power budget. If device geometries are increased and more power is dissipated, analogue systems can be made to perform to higher accuracies, however then the computational efficiency of digital systems tends to be superior.

- *Quantifying* these benefits, it can be shown that the cost (silicon area and power consumption) of analogue computation is exponential with respect to SNR, whereas the cost of digital computation is linear. In addition to this, the starting overhead (at low SNR,) of analogue is low, whereas for digital is high. This sets a trend where the benefits of each method can be divided using SNR alone (see Fig. 2.7 [22, 4]). For lower SNR's, analogue techniques can have many order of magnitude area and power advantage, whereas for higher precision computation digital techniques have the cost advantage.

These conclusions are the result of deriving mathematical expressions to quantify computational cost that are based on the fundamental limits of each technique. Although these provide the ultimate theoretical performance of each representation technique, they do not consider implementation issues, with circuit design and wafer processing being far from ideal. Following are qualitative comparisons of various microelectronic representations in performing common computational tasks; implementation issues being considered.

2.4.2 Linear operations

The most common mathematical computations are in fact linear operations. These include addition, subtraction, multiplication, division and so on. Implementing these computations in different ways can prove hugely beneficial. For example, to add two currents, only a single wire is needed (by Kirchhoff's Current Law,) whereas an 8-bit digital implementation would require 8 full-adder stages, comprising a total of at least 228 transistors. Similarly, a multiplication can be achieved using a Gilbert (translinear) multiplier circuit [24] employing only 8 transistors. Here the equivalent digital solution would be an 8-bit array multiplier

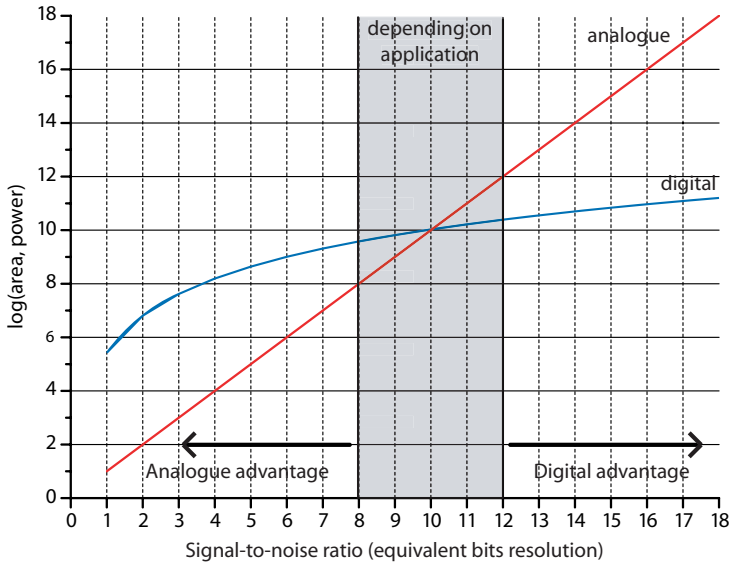


Figure 2.7: The relative cost of computation using analogue or digital signal processing. The crossover between analogue and digital having the advantage is in between 50dB to 72dB SNR (8 to 12 bits resolution) depending on application and circuit topology.

requiring an excess of over 2000 transistors. In these examples silicon area can be saved using analogue techniques, however as always in electronic design, the various trade-offs need be considered. A qualitative comparison of the most popular techniques used for linear arithmetic computation is illustrated in Table 2.2.

For these comparisons, sampled data techniques have been combined with their respective continuous-time counterparts as these are both based on the same underlying circuit theory. To substantiate this Furth et al. [23] have shown these continuous-time and sampled-data techniques to follow similar SNR to power consumption relationships.

2.4.3 Non-linear operations

In most complex processing tasks, the underlying computation tends to be non-linear. This may comprise of an array or bank of linear functions to achieve the overall non-linear behaviour. A qualitative comparison, as previously presented for linear operations, has been formulated for selected common non-linear functions, shown in Table 2.3.

Signal Representation	Topology	Silicon Area	Power	Accuracy	Noise	Speed	Ref.
Addition, Subtraction, Summation							
Current-mode analogue[1]	current addition (KCL)	best	best	good	excellent	good	[25]
Voltage-mode analogue[2]	charge domain (switched-cap)	good	good	good	excellent	good	[26]
Spike domain[3]	logic OR	good	good	excellent	good	excellent	[27]
Digital[4]	parallel counter, ripple adder	fair	good	excellent	excellent	excellent	[28]
Multiplication, Division							
Current-mode analogue	Gilbert multiplier	excellent	excellent	excellent	good	fair	[24]
Voltage-mode analogue	flipped voltage followers	good	good	good	good	fair	[29]
Spike domain	pulse-mode neuron	good	good	poor	good	good	[30]
Digital	array, tree multiplier	poor	fair	excellent	excellent	excellent	[31]
Scaling							
Current-mode analogue	scaled current mirror	excellent	excellent	good	fair	good	-
Voltage-mode analogue	operational amplifier	good	fair	good	good	good	-
Spike domain	shift register counter	good	good	fair	excellent	good	[32]
Digital	barrel shift and accumulate	fair	good	excellent	excellent	excellent	[33]

[1] Provide maximum resource efficiency (area and power) [4].
[2] Provide good all-round performance.
[3] Provide good noise immunity, robustness and resource efficiency [27].
[4] Provide highest speed operation and precision [4].

Table 2.2: A qualitative comparison of linear computations implemented using different signal representation techniques.

Signal Representation	Topology	Silicon Area	Power	Noise	Accuracy	Speed	Ref.
Comparison, Thresholding[1]							
Current-mode analogue	current comparator	excellent	excellent	good	fair	fair	[25]
Voltage-mode analogue	operational amplifier	good	good	good	good	good	-
Spike domain	integrate, logic AND	excellent	good	excellent	fair	excellent	[34]
Digital	subtractor	fair	fair	good	excellent	excellent	[33]
Exponential, Logarithm, Square, Root[2]							
Current-mode analogue	translinear circuits	excellent	excellent	good	good	fair	[35]
Voltage-mode analogue	non-linear V to I	good	fair	good	good	fair	[36]
Spike domain	non-linear neuron	good	good	good	good	good	[34]
Digital	root/division algorithm	fair	fair	excellent	excellent	good	[37]
Filtering, Integration, Differentiation, Fourier Transform[3]							
Current-mode analogue	Log domain	good	excellent	good	excellent	excellent	[38]
Voltage-mode analogue	Charge domain (switched-cap)	good	excellent	fair	good	good	[26]
Spike domain	Lossy integrate & fire neuron	good	good	good	fair	good	[27]
Digital	IIR/FIR filters, FFT	poor	fair	good	excellent	good	[39]

[1] The direct comparison of continuous signals makes analogue comparators the most easily implementable, whereas digital comparison techniques are typically implemented using subtraction driven combinational logic.
[2] Analogue realisations are based on translinear techniques or exploitation of non-linear component response, whereas digital implementations require either ROM-based lookup tables or synthesis of custom arithmetic-logic-unit (ALU) type hardware.
[3] Digital implementation provides better reconfigurability, stability to drift/temperature and low frequency operation.

Table 2.3: A qualitative comparison of non-linear computations implemented using different signal representation techniques.

2.4.4 Hybrid System Organisation

The ultimate goal of using a hybrid approach is to exploit different representation strategies throughout a system; ideally concocting a cocktail of circuit topologies to achieve optimum performance for a given processing task. Unfortunately, we are unable to simply pick-and-match circuit blocks to form a complete system. Conversion techniques must be implemented whenever signal representation changes, however these impose power constraints on the overall system.

Most modern applications typically require both analogue and digital techniques to work along side one another as the bare minimum. Since the real world is analogue, any system requiring a sensor interface requires analogue electronics. On the other hand, as most control systems and communication protocols are digital, any system requiring external interface capability requires digital electronics.

This paradigm itself defines a minimum of one data converter required to be used. Therefore, in order to best utilise resources, it would be best to use this data conversion to our advantage by using this as the main conversion stage within a system. Using previously mentioned signal representation techniques, there exist several architectures that fulfil these criteria (see Fig. 2.8).

2.5 The Technology

For such hybrid processing architectures, CMOS technology is ideally suited. Inherently being a digital process, representation of discrete data is possible in all forms, whether it be, clocked-digital, asynchronous or spike domain. Furthermore, through its wide use in mixed-signal systems, circuit elements well characterised for analogue operation are provided.

This section shall deal with some important aspects in CMOS low power design. For continuous-signal (analogue) design, the weak inversion operating region shall be discussed with emphasis on reliable simulation and noise modelling and device mismatch. Similarly, for discrete-signal design, asynchronous digital implementation issues will be discussed with emphasis on delay modelling, trip-point matching and power reduction techniques.

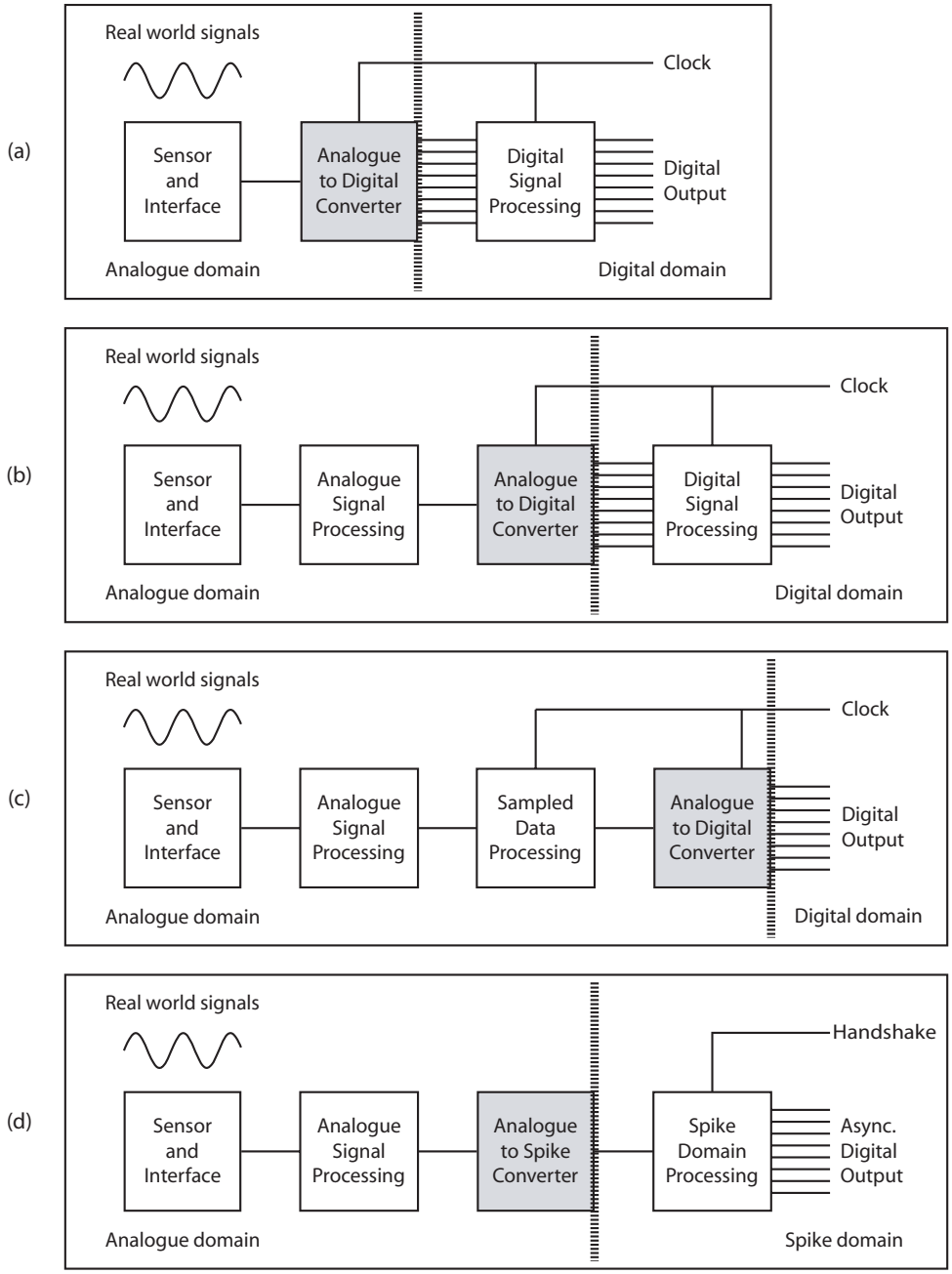


Figure 2.8: Hybrid processing architectures with a single data conversion stage. (a) conventional analogue front-end with digital processor and output (b) hybrid analogue/digital processing platform with digital output (c) hybrid analogue and sampled data processing platform with digital output and (d) hybrid analogue/spike-domain processor with asynchronous digital output

2.5.1 Weak Inversion Technology

It is widely accepted that for low power analogue design, the MOS transistor is most efficiently[1] used in the weak inversion region. Furthermore, the exponential gate-source voltage to drain-current relationship extends translinear circuit theory to be applicable in CMOS, as in bipolar technology. This makes realisation of powerful mathematical operations extremely cost beneficial; both in computational efficiency and silicon area.

The basic model for MOS operation in the weak inversion region [40] is given by:

$$I_{DS} = I_0 \left(\frac{kT}{q}^2\right) e^{\frac{q\kappa V_G}{kT}} \left(e^{\frac{qV_S}{kT}} - e^{\frac{qV_D}{kT}}\right) + G_D V_{DS} \tag{2.1}$$

where I_0 is the pre-exponential constant, V_G is the gate voltage, V_S is source voltage, V_D is drain voltage (all relative to substrate,) κ is the body effect ($\kappa = \partial\psi_S/\partial V_G$, where ψ_S is the surface potential,) k is Boltzmann's constant, T is temperature, q is the electronic charge and $G_D = \partial I/\partial V_D$ is the coefficient for channel length modulation (Early) effect; being different to that for strong-inversion.

This section shall continue by discussing some of the issues that can be particularly detrimental to circuit performance when using devices operating in weak inversion.

Simulation Models

There exist several different MOSFET simulation models; virtually all being valid in strong inversion, however, many of these are simply extrapolated to cover weak inversion operation and therefore provide inaccurate results. A careful review of various simulation models with emphasis on validity and continuity from strong to weak inversion can be found in reference [41].

The two most popular models are the BSIM(V3+) and EKV(V2+) families; both physics-based models being accurate and continuous throughout all operating regions. The BSIM model; an empirically adjusted SPICE model has been adopted (BSIM3v3) as the industry standard CMOS simulation model. On the other hand, the EKV model; built on fundamental charge-based physics has been designed for and dedicated to low power

[1] In weak inversion the gm/I ratio is at a maximum; a measure of how efficiently a transistor uses current to generate transconductance

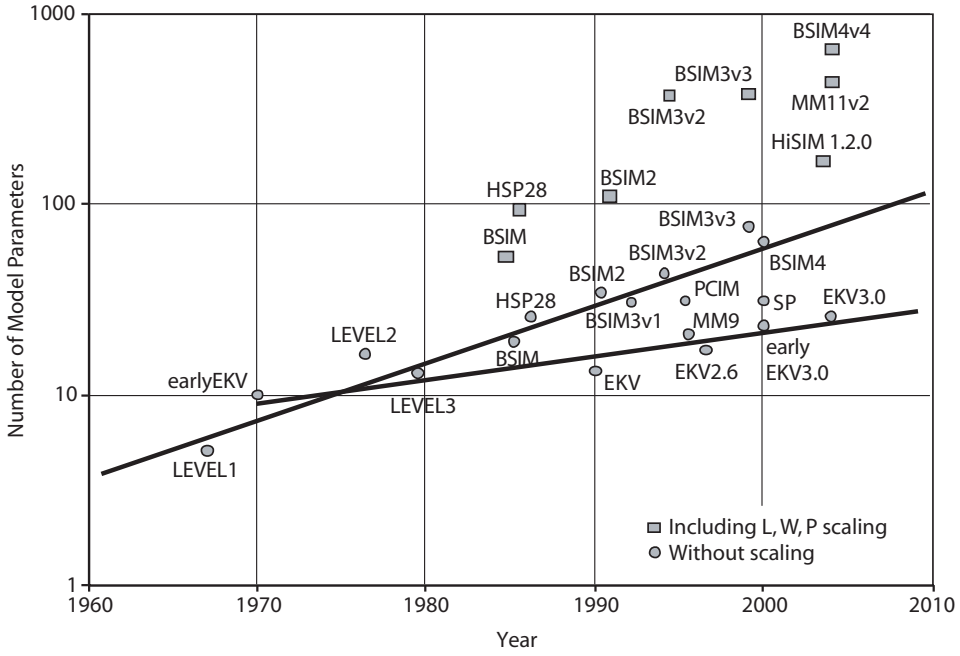


Figure 2.9: Evolving complexity of common MOSFET simulation models; the two plots illustrating the trend for the BSIM and EKV models.

circuit simulation. Most modern simulators are compatible with both models; the most widespread in each case being BSIM3v3.2 and EKV2.6. In comparison, the BSIM model is more complicated, requiring many more parameters than EKV; having only 18 intrinsic parameters (see Fig. 2.9). Furthermore, the EKV model provides a single expression for drain current valid in all modes of operation. For these reasons, the EKV model is often used as an intuitive tool; useful for fast simulation and can be very helpful at understanding device behaviour.

Noise in Weak Inversion

Ultra-low power weak inversion circuits imply low current and/or voltage levels and are therefore more susceptible to the effects of noise. Hence a good understanding in noise in weak inversion MOS transistors would be most useful. Traditional independent noise mechanisms include thermal, shot, flicker, recombination and burst noise [42].

White Noise is the expression given to noise with a flat power spectrum; the most common being thermal or Johnson noise. Such an expression has been derived for subthreshold

MOS transistors by considering the weakly inverted transistor channel to be composed of a series of resistors [43]. Furthermore, Sarpeshkar et al. have developed a single expression for white noise by unifying the processes of thermal and shot-noise for weak inversion MOS transistors [44].

$$\overline{I_{white}{}^2} \simeq 2qI_{sat}\Delta f \tag{2.2}$$

with q being the electronic charge, I_{sat} being the DC current level and Δf being the bandwidth.

Pink Noise describes those noise sources with a power spectrum inversely proportional to frequency; in particular flicker noise [45].

$$\overline{I_{flicker}{}^2} \simeq \frac{KI_{sat}^{p}\Delta f}{WL}\frac{1}{f} \tag{2.3}$$

with K and p being process dependant constants, W and L being the device width and length and f being the frequency.

By comparing (2.2) and (2.3,) it can be concluded that flicker noise dominates for $f < KI/2q$ in weak inversion MOS transistors. Furthermore, measured results of flicker noise in weak inversion transistors [46] [47] tend to suggest that PMOS devices in general are quieter than NMOS. This is due to the fact that different noise mechanisms exist for NMOS and PMOS devices. Flicker noise in NMOS devices is thought to follow carrier density fluctuation, whereas in PMOS devices it is mobility fluctuation and the strong gate bias that are responsible.

MOSFET Matching in Weak Inversion

In CMOS fabrication, there exist two types of wafer processing error to consider. Global variation accounts for the total variation in *absolute* value of a component over a wafer or a batch. On the other hand, local variation reflects the *relative* variation in a component value with reference to an adjacent component on the same chip.

Absolute value variations are usually provided by characterising the process at key points often referred to as *process corners*. In contrast, relative value variations are modelled sta-

tistically, often provided though Monte-Carlo models using gaussian spreads to represent the fluctuating parameters. It is these relative variations, often referred to as *device mismatch* that pose a challenging task to the analogue designer. Furthermore, it is circuits operating in the weak inversion region [48] that are most susceptible to such errors.

Measurements of weak inversion MOS mismatch [48, 49, 50, 51] have identified three main factors effecting device mismatch.

- *The edge effect* causes variations in drain current and is dependant on device position with respect to surrounding structures. For example, in an array of identical transistors, those transistors located at the perimeter will exhibit this edge effect; typically 5-15% for NMOS and 20-50% for PMOS. This can be attributed to various factors. Uneven poly-silicon etch rates at such edge-conditions can alter the effective device size; the etch rate being affected by the etch area. Also doping variations resulting from off-perpendicular ion implantation and diffusion from nearby structures (eg. wells) contribute to this edge effect.
- *The striation effect* (sometimes classed as *quasi-deterministic*) manifests itself as a sinusoidal spatial variation in drain current. The amplitude of this variation is typically 30% the average drain current and the spatial period varies slowly from 100-300μm. This is thought to be due to gas direction flow in wafer processing in implantation chambers.
- *Random variations* manifest themselves to short distance fluctuations in threshold voltage and drain current. Such effects are often attributed to phenomena such as gate oxide non-uniformities (granularity, trapped charge, thickness,) uneven dopant distribution and local effective mobility fluctuations. It is assumed that these physical properties are independent random variables and that the correlation distance of the statistical disturbance is small compared to the active device area. These assumptions lead to characterising the normalised device property distribution with a spatial zero-mean gaussian distribution function.

Threshold Mismatch: The standard deviation of short distance transistor threshold voltage matching, is given by:

$$\sigma(\Delta V_{T0}) = \frac{AV_{T0}}{\sqrt{W.L}} \tag{2.4}$$

with AV_{T0} being a device-specific constant and W.L being the active device width and length, therefore the larger the area, the better the matching due to the averaging of short distance variations.

The dependance of AV_{T0} on physical properties, is given by:

$$AV_{T0} = \frac{qt_{ox}\sqrt{2Nt_{dl}}}{\epsilon_0\epsilon_{ox}} \tag{2.5}$$

with N being the active doping atoms in the depletion layer, t_{ox} and t_{dl} being the oxide and depletion layer thicknesses respectively and $\epsilon_0\epsilon_{ox}$ being the oxide permittivity.

Current Mismatch: The device current mismatch is dependant on the threshold voltage mismatch in addition to the current factor mismatch. A common expression [52] for drain current mismatch is given by:

$$\sigma\left(\frac{\Delta I_D}{I_D}\right) = \sqrt{\frac{4\sigma^2(V_{T0})}{(V_{GS}-V_{T0})^2} + \frac{\sigma^2(\beta)}{\beta^2}} \tag{2.6}$$

with $(V_{GS}-V_{T0})$ being the overdrive voltage and β being the current factor. This expression is normally reduced to a basic area dependance, as in the case for threshold mismatch:

$$\sigma\left(\frac{\Delta I_D}{I_D}\right) = \frac{AI_{Dx}}{\sqrt{W.L}} \tag{2.7}$$

with AI_{Dx} being a device-specific constant, normally quoted in foundry documentation for various values of overdrive voltage.

The current mismatch therefore *increases* with higher transconductance. This has devastating effect on weak inversion circuits; known for having a maximum transconductance for a given current. At the circuit level, it is therefore advantageous to bias any transistors operating in weak inversion at a constant I_{DS} rather than a constant V_{GS}. Biasing at a constant current and thinking in terms of current-domain signals is the essence of current-mode approach in circuit design.

Mismatch Reduction Techniques: Generally these sources of mismatch can be greatly reduced through careful layout techniques. By using symmetric, regular device placements together with use of dummy devices, the edge effect can be completely eliminated. Long-range gradients and striation effects can be reduced using common-centroid layout strategies. Furthermore, random variations can be reduced through increased device sizing. A comprehensive review on careful layout techniques for improved matching is given in reference [53].

Technology Scaling generally has a beneficial effect on device mismatch. It is evident that with decreasing feature sizes the gate oxide thickness decreases and so the oxide quality improves. As the gate oxide thickness is directly proportional to the threshold voltage mismatch (see Eqn. 2.5), this is therefore improved. On the other hand, the current factor mismatch remains more or less constant, hence any improvement in current mismatch is due to better threshold voltage matching.

By collating statistical mismatch data (confidential) over a wide variety of CMOS technologies, the effect on threshold voltage area dependance (mismatch) with minimum technology feature size is established. This trend is illustrated for NMOS and PMOS devices separately in Fig. 2.10. An interesting observation is that although traditionally NMOS devices match better than PMOS, in deep submicron technologies this becomes reversed. This can be explained; not by PMOS matching improving for deep submicron technologies but rather NMOS matching deteriorating. The reason for this is due to the increased doping levels required to maintain acceptable depletion widths in deep submicron. For feature sizes larger than 0.25μm it was sufficient to produce NMOS devices using a relatively low doped p-substrate. However, for sub-250nm technologies it is required to increase basic substrate doping by placing NMOS devices in p-wells, similarly to the way PMOS devices have been traditionally placed in n-wells.

2.5.2 Asynchronous Technology

The clocked synchronous paradigm is currently the dominant design methodology for digital systems. While this approach has proved hugely successful over many decades, limitations and drawbacks do exist. For example, the requirement in distributing high-speed clocks over a large chip area, with precision, is complex and the clock tree itself dissipates a significant proportion of the total power consumption. Also, clocking an entire system does not make the most efficient use of resources; for circuits not contributing to a particular process still

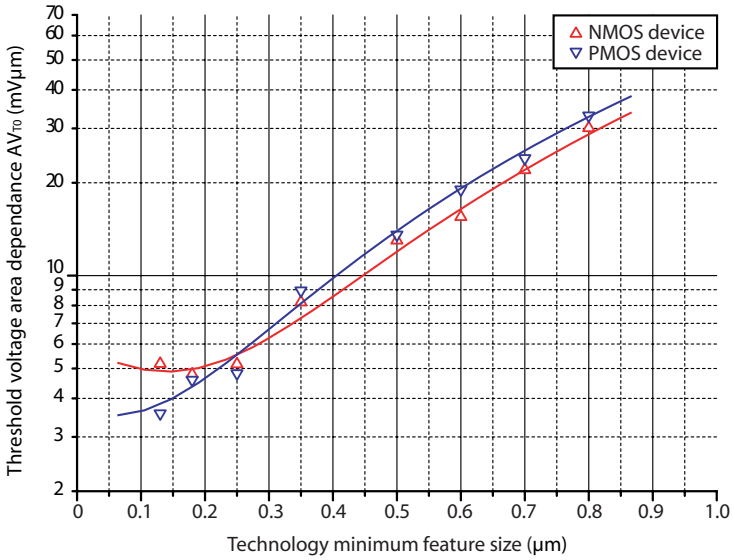


Figure 2.10: Effect of CMOS technology scaling on threshold voltage mismatch

burden the power supply. Furthermore, in mixed-signal systems, digital clocks often pickup in sensitive analogue circuitry resulting in degraded performance. These problems become more severe as device sizes continue to shrink and as clock frequencies continue to rise.

Asynchronous design offers an alternative to the clocked system methodology. This overcomes the above limitations by dispensing with the clock and using self-timed signaling to control the sequencing of computations in the system. Such circuits have the potential for very low power consumption, as only the parts of the circuit that are utilised at any time have switching activity and thus consume power.

In order to utilise such techniques effectively, a good understanding of transistor-level logic implementation is timely. This section shall discuss some key issues in logic design, particularly focusing on power dissipation and delay modelling.

Power Consumption

CMOS logic is typically classed as *static logic,* ideally consuming no power for no activity. In reality however, power dissipation can be attributed to three main sources [33]. These are:

- *Static Dissipation* refers to all the sources that draw constant current from the power

supply. In static CMOS logic, this is due to reverse bias leakage between diffusion regions and the substrate in addition to subthreshold conduction.

$$P_{static} = (I_{rb} + I_{sub})V_{DD} \tag{2.8}$$

with I_{rb} being the reverse bias leakage, I_{sub} being the subthreshold (diffusion) current and V_{DD} being the supply voltage.

- *Dynamic Dissipation* refers to power used that is directly related to activity, i.e. the more activity, the more dynamic power. This is due to the charging and discharging of load capacitances during switching. The expression for dynamic switching power is given by:

$$P_{dynamic} = \frac{C_L V_{DD}^2}{t_p} \tag{2.9}$$

with C_L being the load capacitance and t_p being the minimum switching interval.

- *Short-Circuit Dissipation* is a form of dynamic dissipation; this occurs during transitions when both NMOS and PMOS devices are conducting and in saturation and a direct route exists for current to flow between the power rails. The expression for short-circuit dissipation is given by: [33]

$$P_{short-circuit} = \frac{\beta}{12}(V_{DD} - 2V_T)^3 \frac{t_{edge}}{t_p} \tag{2.10}$$

with β being the transconductance coefficient, V_T being the thermal voltage and t_{edge} being the edge rise or fall time (assuming that the rising and falling edges are the same.)

The total power dissipation can therefore be determined by considering the sum of equations (2.8), (2.9) and (2.10). However, in complex circuit design, it is impractical to evaluate the above at each individual logic node. Since for any well designed digital logic, the dynamic dissipation will be the predominant factor, a simple approximation can be made. By lumping together all the capacitance driven by gate outputs, the following expression is formed:

$$P_{approx} = \frac{\delta C_{total} V_{DD}^2}{t_p} \tag{2.11}$$

with δ being the percentage activity and C_{total} being the total load capacitance.

Delay Modelling

In asynchronous circuit design, a crucial tool in creating reliable self-timed circuits is delay modelling. As no clock exists to synchronise and condition signals, the focus is on building well balanced circuits to provide glitch-free operation. In order to achieve this, a strong understanding of switching characteristics is paramount.

Inverter Timing Analysis

The switching speed of any CMOS gate is limited by the time taken to charge and discharge the load capacitance. In order to establish this, it is useful to analyse the constituent timings within a simple inverter.

The *fall-time* of a CMOS inverter is dictated by the NMOS switching characteristic. This is defined as the time taken for a waveform to fall from 90% to 10% of its steady-state value. The analytical expression [33] is given by:

$$t_f = \frac{2C_L}{\beta_n V_{DD}(1-n)}\left[\frac{(n-0.1)}{(1-n)} + \frac{ln(19-20n)}{2}\right] \simeq \frac{kC_L}{\beta_n V_{DD}} \tag{2.12}$$

with β_n being the transconductance coefficient for a NMOS device, n being the NMOS turn-ON point ($n = v_{tn}/V_{DD}$ with v_{tn} being the NMOS threshold voltage) and k being the switching strength parameter; lumping together all the n terms. The assumption is then made that the k parameter is the same for NMOS and PMOS devices.

Similarly, the *rise-time* of a CMOS inverter is dictated by the PMOS switching characteristic. This is defined as the time taken for a waveform to rise from 10% to 90% of its steady-state value. The analytical expression [33] is given by:

$$t_r = \frac{2C_L}{\beta_p V_{DD}(1-p)}\left[\frac{(p-0.1)}{(1-p)} + \frac{ln(19-20p)}{2}\right] \simeq \frac{k_p C_L}{\beta_p V_{DD}} \tag{2.13}$$

with β_p being the transconductance coefficient for a PMOS device, n being the PMOS turn-ON point ($p = v_{tp}/V_{DD}$ with v_{tp} being the PMOS threshold voltage) and k being the switching strength constant.

Thus for equally sized N- and P-MOS transistors, with $\beta_n = 2\beta_p$ (due to different carrier mobilities,) the rise and fall times are scaled proportionally, i.e. $t_f = t_r/2$. For well balanced inverters it is therefore necessary to design the PMOS with increased aspect ratio.

The *delay-time* is then defined as the time difference between input transition (50%) and the 50% output level. This can be approximated as half the rise or fall time, depending on whether the transition is rising or falling.

Gate Delays

CMOS logic delays are often approximated by reducing a gate to an equivalent inverter. For example, a 3-input NOR gate would reduce to three series PMOS devices for determining the rise time and a single NMOS device for determining the maximum fall time.

In general, the fall time t_f is at_f for a NMOS transistors in series and the rise time t_r is bt_r for b PMOS transistors in series. Similarly, the minimum fall time is t_f/x for x NMOS transistors in parallel and the minimum rise time is t_r/y for y PMOS transistors in parallel. The maximum fall and rise times for parallel devices being achieved when only a single device is contributing to the logic action. Such timings are invaluable for designing circuits that have critical-delay paths, to avoid glitches and hazards. For other methods of determining delay approximations see references [54] and [55].

2.6 Summary

In this chapter neurobiology has been reviewed from an engineering perspective, examining organisational and representation techniques. Extrapolating to microelectronics, these design principles have been used in determining methods suitable for realising electronic systems based on biology. The motivation being that this will lead to development of effective, efficient and robust perceptive systems. It has been established that hybrid structures are a *requisite* in modern sensor processing applications and representation-based system-level design techniques can assist in improving computational efficiency. Weak inversion analogue and asynchronous digital electronics have been identified as the most suitable techniques for biologically-inspired circuit design. Specific implementation issues have been discussed and analysed for low power and robust design methodology. For weak inversion operation, noise and matching analysis suggest that PMOS devices perform better than NMOS in deep submicron technologies. Furthermore, an understanding behind device mismatch provides the knowhow in designing reliably manufacturable subthreshold circuits. For asynchronous logic design, delay modelling and power dissipation analysis provides an intuitive design methodology for realising reliable self-timed and well-balanced circuits with minimal power losses.

References

[1] C. A. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley, 1989.

[2] B. A. Wandell, *Foundations of Vision*. Sinauer Associates, 1995.

[3] J. G. Nicholls, A. R. Martin and B. G. Wallace, *From Neuron to Brain*, ch. Analysis of signals in the nervious system. Sinauer Associates, 1992.

[4] R. Sarpeshkar, *Efficient Precise Computation with Noisy Components: Extrapolation from an Electronic Cochlea to the Brain*. PhD thesis, California Institute of Technology, Pasadena, California, 1997.

[5] W. Gerstner, W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[6] L. W. Swanson, *Brain Architecture: Understanding the Basic Plan*. Oxford University Press, 2002.

[7] G. M. Shepherd (ed.), *The Synaptic Organization of the Brain*. Oxford University Press, 1998.

[8] J. G. Nicholls, A. R. Martin and B. G. Wallace, *From Neuron to Brain*. Sinauer Associates, 1992.

[9] G. M. Shepherd, "Microcircuits in the nervous system," *Scientific American*, vol. 238, no. 2, pp. 93–103, 1978.

[10] C. Koch, T. Poggio and V. Torre, "Nonlinear interactions in a dendritic tree: Localization, timing and the role of Information Processing," *Proceedings in the National Academy of Science USA*, vol. 80, no. 9, pp. 2799–2802, 1983.

[11] P. Rakic, "Local Circuit Neurons," *Neuroscience Research Program Bulletin*, vol. 13, no. 3, pp. 295–416, 1975.

[12] K. A. Boahen, "A Retinomorphic Chip with Parallel Pathways: Encoding ON, OFF, INCREASING, and DECREASING Visual Signals," *Kluwer Analog Integrated Circuits and Signal Processing*, vol. 30, no. 2, pp. 121–135, 2002.

[13] K. A. Zaghloul and K. A. Boahen, "Optic Nerve Signals in a Neuromorphic Chip I: Outer and Inner Retina Models," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 4, pp. 657–666, 2004.

[14] P. Sterling, "How retinal circuits optimize the transfer of visual information," *The Visual Neurosciences, by L. M. Chalupa and J. S. Werner, eds.*, pp. 234–259, 2004.

[15] R. Gregorian and G. C. Temes, *Analog MOS Integrated Circuits for Signal Processing*. John Wiley & Sons, 1986.

[16] C. Toumazou, J. B. C. Hughes and N. C. Battersby, eds., *Switched-currents: An Analogue Technique for Digital Technology*. IEE Publishing, 1993.

[17] C. F. Moss and J. A. Simmons, "Acoustic image representation of a point target in the bat Eptesicus fuscus: Evidence for sensitivity to echo phase in bat sonar," *Journal of the Acoustical Society of America*, vol. 93, pp. 1553–1562, 1993.

[18] P. A. Cariani, "Temporal coding of sensory information in the brain," *Acoustical Sciency and Technology: The Acoustical Society of Japan*, vol. 22, no. 2, pp. 77–84, 2001.

[19] D. H. Hubel, "Receptive Fields of Single Neurons in the Cat's Striate Cortex," *Journal of Physiology (London)*, vol. 148, pp. 574–591, 1959.

[20] V. B. Mountcastle, "Modality and Topographic Properties of Single Neurons of Cat's Somatosensory Cortex," *Journal of Neurophysiology*, vol. 20, pp. 408–434, 1957.

[21] B. J. Hosticka, "Performance Comparison of Analog and Digital Circuits," *Proceedings of the IEEE*, vol. 73, no. 1, pp. 25–29, 1985.

[22] E. A. Vittoz, "Future of analog in the VLSI environment," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 1372–1375, 1990.

[23] P. M. Furth and A. G. Andreou, "Bit-energy comparison of discrete and continuous signal representations at the circuit level," *4th Workshop on Physics and Computation*, 1996.

[24] B. Gilbert, "A precise four-quadrant multiplier with subnanosecond response," *IEEE Journal of Solid-State Circuits*, vol. SC-3, p. 365, 1968.

[25] C. Toumazou, F. J. Lidgey and D. G. Haigh, *Analog IC Design: The Current-Mode Approach*. London: Peter Perigrinus, 1990.

[26] P. E. Allen, E. S. Sinencio and E. Sanchez-Sinencio, *Switched Capacitor Circuits*. Kluwer Academic Publishers, 1984.

[27] A. F. Murray, D. Del Corso and L. Tarassenko, "Pulse-stream VLSI neural networks mixing analog and digital techniques," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 193–204, 1991.

[28] D. J. Kinniment, J. D. Garside and B. Gao, "A comparison of power consumption in some CMOS adder circuits," *Proceedings of PATMOS, C. Piguet and W. Nebel, eds.*, pp. 119–132, 1995.

[29] J. Ramirez-Angulo, S. Thoutam, A. Lopez-Martin, R. J. Carvajal, "Low-voltage CMOS analog four quadrant multiplier based on flipped voltage followers," *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 681–684, 2004.

[30] H. Hikawa, "A new digital pulse-mode neuron with adjustable activation function," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 236–242, 2003.

[31] J. H. Satyanarayana and K. K. Parhi, "A theoretical approach to estimation of bounds on power consumption in digital multipliers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 6, pp. 473–481, 1997.

[32] H. Hikawa, "Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 545–553, 1999.

[33] N. H. E. Weste and K. Eshraghian, *Priciples of CMOS VLSI design: A systems perspective*. Addison-Wesley, 1993.

[34] A. F. Murray, "Pulse arithmetic in VLSI neural networks," *IEEE Micro*, vol. 9, no. 6, pp. 64–74, 1989.

[35] B. Gilbert, *Analog IC Design: The Current-Mode Approach*, ch. Current-mode circuits from a translinear viewpoint: A tutorial. Peter Peregrinus Ltd., 1990.

[36] D. Quoc-Hoang Duong, N. Trung-Kien Nguyen and L. Sang-Gug, "Ultra low-voltage low-power exponential voltage-mode circuit with tunable output range," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 729–732, 2004.

[37] M. D. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic Publisher, 1994.

[38] E. M. Drakakis, A. J. Payne and C. Toumazou, "Log-domain filtering and the Bernoulli cell," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, no. 5, pp. 559–571, 1999.

[39] J. Proakis, D. Manolakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*. Pearson US Imports & PHIPEs, 1995.

[40] M. D. Godfrey, "CMOS device modeling for subthreshold circuits," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 8, pp. 532–539, 1992.

[41] J. Georgiou, *Micropower Electronics for Neural Prosthetics*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, 2002.

[42] W. M. Leach, "Fundamentals of low-noise analog circuit design," *Proceedings of the IEEE*, vol. 82, pp. 1515–1538, 1994.

[43] C. Enz, *High Precision CMOS Micropower Amplifiers*. PhD thesis, Ecole Polytechnique Federale de Lausanne, Switzerland, 1990.

[44] R. Sarpeshkar, T. Delbruck, and C. A. Mead, "White noise in MOS transistors and resistors," *IEEE Circuits and Devices Magazine*, vol. 9, no. 6, pp. 23–29, 1993.

[45] C. Schutte and P. Rademeyer, "Subthreshold 1/f noise measurements in MOS transistors aimed at optimizing focal plane array signal processing," *Kluwer Analog Integrated Circuits and Signal Processing*, vol. 2, pp. 171–177, 1992.

[46] T. Delbruck, *Investigations of visual transduction and motion processing*. PhD thesis, California Institute of Technology, Pasadena, California, 1993.

[47] J. Chang, A. A. Abidi and C. R. Viswanathan, "Flicker noise in CMOS transistors from subthreshold to strong inversion at various temperatures," *IEEE Transactions on Electron Devices*, vol. 41, no. 11, pp. 1965–1971, 1994.

[48] A. Pavasovic, *Subthreshold region MOSFET mismatch analysis and modeling for analog VLSI systems.* PhD thesis, John Hopkins University, Baltimore MD, 1990.

[49] A. Pavasovic, A. G. Andreou, and C. R. Westgate, "Characterization of Subthreshold MOS Mismatch in Transistors for VLSI Systems," *Kluwer Journal of VLSI Signal Processing*, vol. 8, pp. 75–85, 1994.

[50] F. Forti and M. E. Wright, "Measurement of MOS current mismatch in the weak inversion region," *IEEE Journal on Solid State Circuits*, vol. 29, no. 2, pp. 138–142, 1994.

[51] N. Kumar, P. O. Pouliquen and A. G. Andreou, "Device mismatch limitations on performance of a Hamming distance classifier," *IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 327–334, 1993.

[52] M. Pelgrom, A. Duinmaijer and A. Welbers, "Matching Properties of MOS transistors," *IEEE Journal on Solid State Circuits*, vol. 24, pp. 1433–1440, 1989.

[53] R. A. Hastings, *The Art of Analog Layout.* Prentice Hall, 2000.

[54] T. Sakurai and A. R. Newton, "Delay analysis of series-connected MOSFET circuits," *IEEE Journal of Solid State Circuits*, vol. 26, no. 2, pp. 122–131, 1991.

[55] S. Dhar and M. A. Franklin, "Optimum buffer circuits for driving long uniform lines," *IEEE Journal of Solid State Circuits*, vol. 26, no. 1, pp. 32–40, 1991.

Chapter 3

Modern Vision Processing Technology

3.1 Introduction

Computer and machine vision are disciplines that have matured substantially in the past two decades. Computer vision consists of image processing (image in, image out), image analysis (image in, measurement out) and image understanding (image in, high-level description out). Machine vision is the application of computer vision to enable a system to detect or extract and subsequently act upon a certain visual feature in a control or analysis task.

Machine vision technology is becoming an increasingly important and in some cases indispensable tool, for example in manufacturing automation. Applications appear in many disciplines and industries, including: semiconductor, electronics, pharmaceuticals, packaging, medical devices, biomedical, space, automotive, security, surveillance and consumer goods. Machine vision systems offer a non-contact means of inspecting and identifying parts, accurately measuring dimensions, or guiding robots, instruments or other machines during positioning, navigation or stabilising operations.

Although such techniques continue to enjoy much success, effective realisation of perceptive vision processing is a function that continues to elude us. Visual perception is a high level process often associated with the human visual system. This is because, biological vision is undoubtedly superior-to and unrivaled-by any synthetic artifact in perceptive processing. For example, through our visual perception we immediately perceive a room as containing windows, furniture and objects without delaying to process or analyse the image.

Such neurobiological visual perception is facilitated by filtering, extracting and converting the incident image through a chain of sub-processes of increased abstraction and complexity starting at the ocular optics and retina. The retina preconditions, amplifies, compresses and extracts parallel visual streams from the incident image. This set of neural images convey various features via the Lateral Geniculate Nucleus (LGN) to the primary visual cortex for more specific feature extraction (eg. orientation selective). As subsequently information is conveyed to higher neural layers, the amount (or quantity) of data reduces, whereas the perceptive level (or quality) increases.

Conventional computer vision processing systems are designed to operate under well controlled conditions for example, with uniform lighting and well-defined targets. Visual perception functions however are not influenced by such factors. For example, an image recognition task should not be affected by lighting, orientation or size. Therefore it would be useful to understand and use the underlying biological principles to produce effective and robust perceptive vision processors.

This chapter begins by reviewing standard silicon process technologies suitable for imaging, image processing and vision processing. Emphasis is given to identify a technology suitable for implementation biologically inspired processing; integrating phototransduction devices with electronics. Sequential and distributed processing architectures are then examined identifying their merits and limitations. Finally, existing techniques (both software and distributed hardware) for centre-of-mass processing and target tracking are reviewed and discussed, and vision chips developed in this field are compared. This task (centroid processing) is considered a high-level task; for it requires image conditioning and pre-processing, object segmentation and beyond and is therefore classed as a perceptive vision process.

3.2 Imager Technology

George Smith and Willard Boyle invented the Charge-Coupled Device (CCD) [1] at Bell Labs in the late 1960's. They were attempting to create a new kind of semiconductor memory for computers. By 1970, the Bell Labs researchers had built the CCD into the world's first solid state video camera. Until the recent invention of the CMOS Active Pixel Sensor (APS) by Eric Fossum in the early 1990's, solid-state imaging systems have relied on Charge-Coupled Device (CCD) technology for the image sensor component.

For the past decade, the CMOS APS has attempted to address some of the weaknesses

of CCD technology. CCD's rely on perfect charge transfer which makes the technology intrinsically radiation "soft" in comparison to CMOS and therefore less straight-forward to use at low temperatures. Furthermore, the difficulty to integrate with on-chip electronics makes high frame-rate imagers and low power operation extremely challenging for CCD technology. Also, comparing a specialist CCD imager process to standard CMOS technology; by far being the most common and highest yielding process, it follows that CMOS imagers would boast a significant cost advantage.

This section is dedicated to these two solid-state imager technologies. By outlining the principles of operation and comparing state-of-the-art CCD and CMOS image sensors, the strengths and weaknesses of each method are established and stated.

3.2.1 CCD Imagers

CCD technology uses the inherent photoresponsiveness of silicon to generate electron-hole pairs on photon absorption. Impurity implants patterned into the silicon together with a suitable voltage bias confine the photo-generated electrons to discrete packets. These charge packets are then transferred to the conversion hardware using various strategies.

There typically exist three main architectures for CCD-based imagers: full-frame, frame transfer and interline. Other techniques (not discussed) include Frame-Interline Transfer, Accordian, Charge Injection and MOS XY Addressable.

- *Full-Frame (FF):* These typically are have the simplest architecture and are therefore the easiest to design, fabricate and operate. In these, the total area of the CCD (100% fill factor) is available for detecting incoming photons during the exposure period. During the readout phase, charge is shifted sequentially across the array therefore necessitating a mechanical shutter to maintain image integrity (i.e. to prevent smearing). This architecture consists of a parallel CCD shift register, a serial CCD shift register and a charge conversion output amplifier. These simplicity of the FF design yields CCD imagers with the highest resolutions and densities.

- *Frame Transfer (FT):* This architecture are very similar to the FF technique, differentiating in the fact that it uses two CCD arrays; one photosensitive and one used exclusively for storage. The idea is to quickly shift the captured scene from the photosensitive region to the storage array. Readout off-chip from the storage register is then

performed similarly to the FF device, whilst the next image is being formed on the sensing array. The advantage is that continuous (shutterless) operation is achievable resulting in faster frame rates. This is however at the expense of image quality (due to smearing), reduced resolution and higher cost.

- *Interline (IL):* These have been devised to address the shortcomings of FT devices. This is achieved by separating the photo-detecting and readout functions by forming isolated photosensitive regions interweaved with lines of light-shielded parallel readout CCDs. After integrating a scene, the signal collected in every pixel is shifted, all in parallel, to the charge storage parallel CCD. Readout and output is then performed similarly to FF and FT devices. This architecture significantly reduces smearing and increases frame-rate. However, the increased pixel complexity and reduced fill factor typically result in higher unit costs and lower sensitivities.

CCD technology has several general advantages and disadvantages. Using a single output amplifier to convert and amplify the charge means excellent global image uniformity. Furthermore, a single output amplifier can be optimised for low-noise operation and combined with a low-noise substrate, CCD imagers boast good dynamic range. On the other hand, as the image is formed through accumulating charge in adjacent wells, an "overfill" would result in charge spilling out to adjacent pockets, referred to as *blooming*. It is technically feasible but not economic to use the CCD manufacturing process to integrate other image sensor requisites, such as the clock drivers, timing logic, and signal processing on the same chip as the photodetectors. These are normally put on separate chips so CCD cameras contain multiple chips. The complete CCD imager architecture (FF organisation) is illustrated in Figure 3.1.

3.2.2 CMOS Imagers

In the past, CMOS imagers based on the *Passive Pixel Sensor (PPS)* architecture suffered from poor image uniformity/dynamic range and limited output bandwidth (due to high readout capacitance). A solution to this problem would be to introduce local gain within the pixel to buffer the signal. However, in doing so the active photo-sensor area to total pixel size ratio (surface fill factor) would be reduced, consequently also reducing the responsivity (incident light power to photocurrent ratio.)

In the past, as minimum transistor feature sizes were multiple-micron, an active sensor

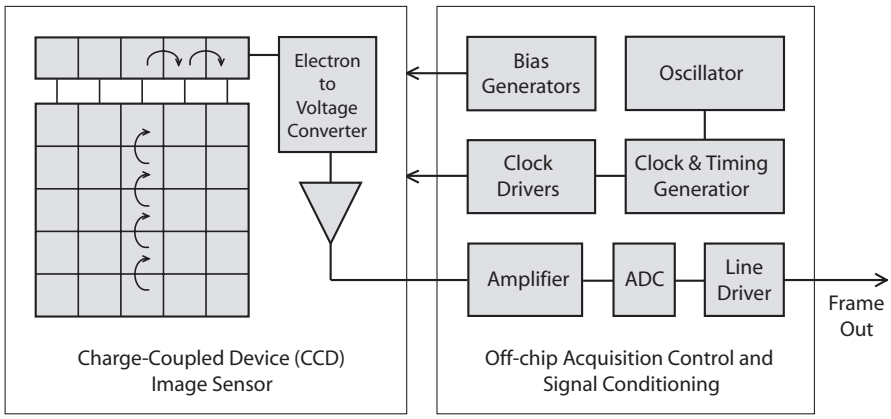


Figure 3.1: A typical CCD imager (full-frame based) architecture.

approach would render an prohibitively low fill factor, resulting in an unusable pixel format. Through CMOS technology scaling, the *Active Pixel Sensor (APS)* [2] was recently made feasible as the ratio of electronic device to photodetecting element has been significantly reduced. The APS approach is to include an in-pixel amplifier performing the charge-to-voltage conversion inside each and every pixel. Using a column and row selection matrix, the individual pixel voltages are scanned off the array and buffered by the column amplifiers.

A drawback of the APS approach is that non-uniformities in the in-pixel amplifiers constitute the so-called *Fixed Pattern Noise (FPN)*. However, a major benefit in using CMOS technology is the ability to integrate photodetecting elements with electronics on the same chip. Subsequently, advanced signal conditioning techniques can be implemented to dramatically reduce the FPN (eg. correlated-double sampling) and further process the image. The complete CMOS imager architecture is illustrated in Figure 3.2.

3.2.3 CCD vs. CMOS

Comparing CCD to CMOS imager technologies [3] [4] several key differences can be found. Table 3.1 makes such a comparison in various performance criteria including: dynamic range, responsivity, uniformity, speed, power and cost [3]. Considering these advantages and disadvantages of CCD and CMOS imagers, it is evident these will continue to co-exist having a complimentary application space:

- CCD image sensors offer superior image quality and flexibility at the expense of system size, cost and power. This technology remains the most suitable for high-end imaging

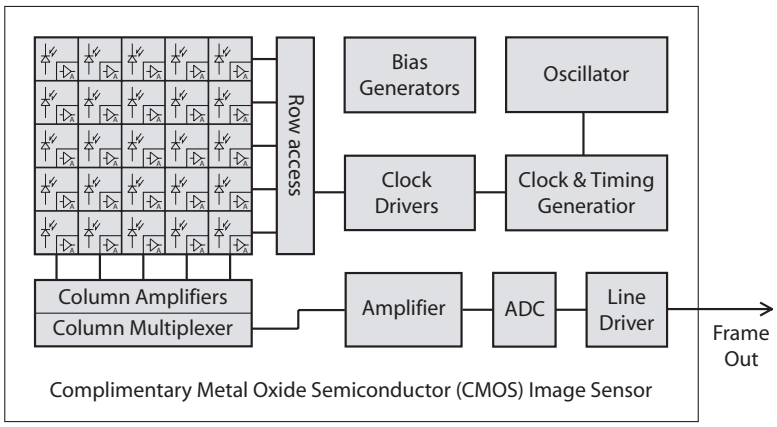


Figure 3.2: A typical CMOS imager architecture

applications. Such examples include: high definition video and still cameras and most high quality industrial, scientific and technical applications.

- CMOS imagers offer superior integration, fabrication cost, power dissipation and system size at the expense of image quality. This technology is the most suitable for high-volume, space- and power-constrained applications with relaxed image quality requirements. Such examples include: security, surveillance, biomedical, biometric, automotive, healthcare and machine vision applications.
- Recent trends in CMOS imager technology have shown potential for APS systems applied to high definition applications traditionally catered for exclusively by CCD systems [5, 6].

3.3 Vision Processing Techniques

Image and vision processing has traditionally involved a modular organisation, consisting of a stand-alone camera, computer interface and PC. Recent developments in processing hardware has enabled embedded processors to substitute the traditional computing platform. Although this presents a more compact, power efficient system, the underlying principle of organisation remains the same; very much a sequential von Neumann based architecture.

This section outlines the strengths and weaknesses of this approach and continues by introducing the distributed processing approach, recently made feasible by advances in microelectronic technology, in particular concerning submicron CMOS.

Attribute	Comparison	Advantage
Responsivity	Both CCD and CMOS have similar responsivities (as both can now include in-pixel gain).	Same
Dynamic Range	CCD achieves superior dynamic range due to lower noise (in substrate and amplifiers).	CCD
Fill Factor	CCD approaches 100% fill factor whereas CMOS typically achieves 20-50%, however this can be greatly improved by using micro-lenses.	CCD
Pixel Size	Full-frame CCD's can have the minimum pixel pitch, due to no in-pixel circuitry.	CCD
Uniformity	CCD has excellent image uniformity due to single output amplifier whereas in CMOS, FPN degrades uniformity hence additional electronics are required for compensation.	CCD
Shuttering	Interline CCD's have the ability to shutter arbitrarily whereas CMOS requires additional electronics.	CCD
Speed	CMOS capable of much higher speed of operation than CCD due to the integration of all functions on the same chip (less capacitance).	CMOS
Windowing	On-chip electronics in CMOS can produce control signals for windowing functionality.	CMOS
Anti-blooming	CMOS has inherent immunity to blooming whereas to reduce in CCD requires alteration to the standard CCD fabrication process.	CMOS
Anti-smearing	CMOS has inherent immunity to smearing whereas this places design constraints in CCD.	CMOS
Biasing & Clocking	CMOS devices operate from a single voltage bias and clock level, generated on-chip whereas CCD require several higher voltage biases.	CMOS
System Size	Electronics integration on single chip in CMOS results in reduced system size compared to CCD.	CMOS
Power	CMOS technology is inherently lower power than CCD. Optimal modular design in CCD for high-speed systems can provide good power efficiency.	CMOS
Cost	Lower total cost in CMOS technology due to wider availability of CMOS fabrication and single-chip implementation.	CMOS

Table 3.1: Comparison of CCD and CMOS imager technologies

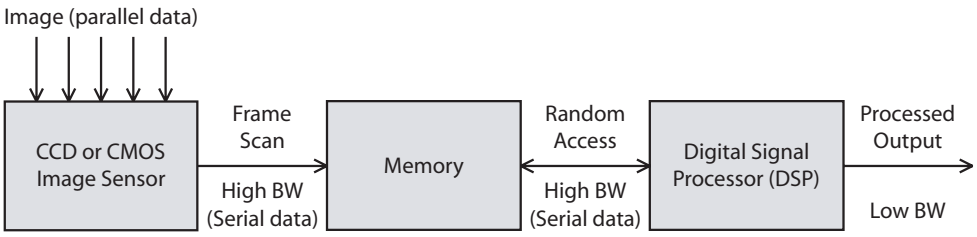


Figure 3.3: A conventional real-time image processing platform

3.3.1 The Modular Approach

This conventional approach uses a standard CCD or CMOS camera (image sensor including all internal image conditioning and pre-processing hardware) for image capture, subsequently transmitting the image to be stored in memory. The processor, either an embedded DSP/FPGA or a conventional PC platform then executes an image processing algorithm, either randomly or sequentially accessing the stored image within memory. This process is typically repeated for each frame in static image processing. If transient image properties are to be processed then a history of previous frames need to be stored in memory. The great merit of this technique is reconfigurability and versatility, for the processing algorithm is defined in software and therefore reprogrammable. This modular architecture for implementing real-time image processing is illustrated in Figure 3.3.

Generally with modern processing hardware, "real-time" processing at standard scan rates (eg. 25Hz) is achievable for many tasks, at the expense of power consumption and system size. This poses a major issue in portable battery-operated machine vision applications having stringent power and size constraints. Furthermore, the sequential processing and communication nature of this technique render it unusable for high frame rate applications. For example, for 8-Bit VGA resolution (640x480 pixel) image at 1000 Frames Per Second (FPS) refresh, the imager alone would require a 2.46Gbps communication bandwidth! It is here the bottleneck of the sequential von Neuman based computational paradigm presents itself.

iCount: A Modular Real-time Image Processing Example

A commercial system implemented using such a platform is iCount by Safehouse Technology Ltd [7]. The product brochure provides the following description:

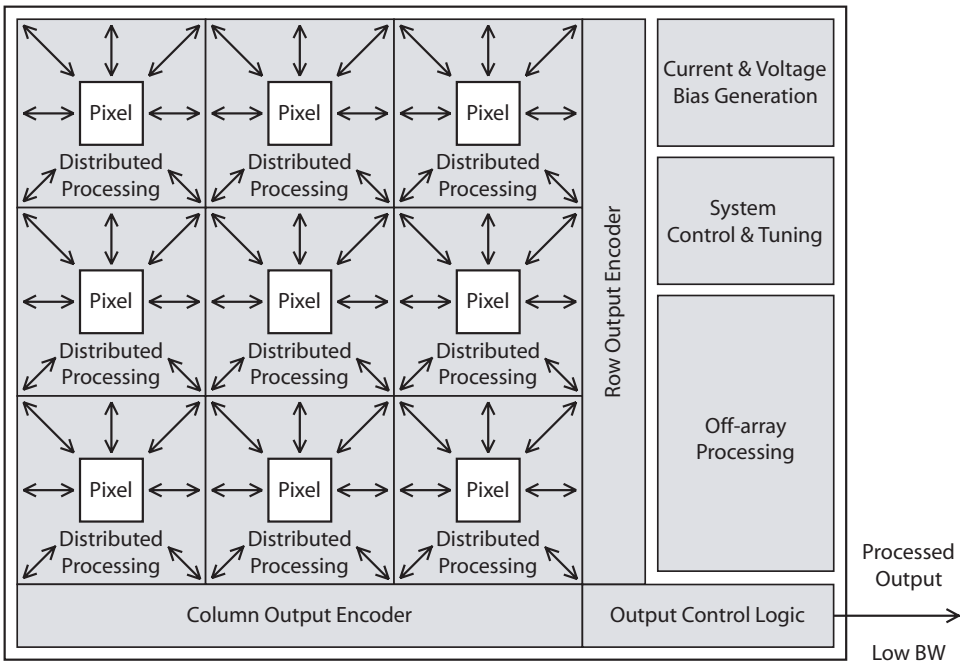


Figure 3.4: Real-time distributed-processing vision chip architecture

"iCount unobtrusively provides counting statistics for people and vehicles moving between user-defined areas. The system uses analogue or digital cameras, runs on standard PC hardware and integrates easily with existing IT infrastructures."

In this application, the modular approach provides a good, usable solution; for there are no power or size constraints and reconfigurability is necessary to set up the various system parameters. For example, in a particular scene object size and shape, boundary crossings, forbidden regions and lighting conditions need to be set.

3.3.2 The Distributed Approach

Distributed or *focal-plane* vision processing has mainly evolved and been developed within the *Bio-inspired Electronics* community in the past 15 years, since Carver Mead introduced the notion of Neuromorphic Electronics[8]. This is due to the fact that distributed vision processing *at least* shares two fundamental properties with neurobiology: massive parallelism and being constructed from basic and identical processing elements.

A distributed processing architecture (see Fig. 3.4) uses electronics embedded within the

photo-detection array, within each pixel; performing *local*, *parallel* and *distributed* processing. Every row and column shares a common output bus for extraction of processed data. Since the output is high-level processed data, the output buses are event- or data-driven; delivering an event or value only when the local processing elements flag a useful result is awaiting. This phenomenally reduces the communication bandwidth, thus a simple asynchronous handshake is the preferred method for off-chip communication. Some such vision systems contain an *off-array processing core* for some post-processing of the sequentially extracted data. Distributed processing vision architectures are often called *vision chips* because all the required functionality is on a single chip, including bias and reference generators, clocks[1], control and tuning logic are contained on a single chip. This has been made possible through use of modern CMOS technology.

The distributed processing paradigm overcomes the bottleneck presented in sequential processing architectures by employing massive parallelism of low speed processing elements[2]. Furthermore, by optimising these elements using hybrid (analogue, digital and spike-domain) circuit topologies, phenomenally good computational efficiency is achievable in real-time vision processors. This has been until recently [9, 10] at the expense of development time and reconfigurability. As vision chips are application specific and hardware based, generally they require more time to design and once fabricated are typically dedicated to a specific algorithm or processing task.

ACE16K: A Distributed Real-time Vision Processing Example

A commercial system implemented using such a platform is the ACE16K [11] by Anafocus Ltd. The product brochure provides the following description:

"A digitally-controlled analog array processor designed for fast image processing applications. Its revolutionary processing capabilities rely on a combined spatial distribution of sensing, processing and storage on an array of identical programmable units."

This product is a generic high frame-rate (surpassing 1000FPS) vision processing system targeted at applications such as textile fault detection, rail inspection from high-speed trains, detection of debris particles in oil flow, high speed inspection during production, etc. The system uses a distributed *Cellular Neural Network (CNN)* architecture to provide the computational power required to process the vast amount of image data. However, the

[1] Use of clocks is normally avoided (for noise and power reasons) unless absolutely critical.
[2] An important advantage of this technique is the processing time is *independent* of array size

main limitations of this approach is the relatively large pixel size and prohibitively high power consumption, excluding it from use in applications with stringent power constraints or those requiring high resolution images.

3.3.3 The Computation-on-Readout Approach

Although the distributed approach is ideal for fast and efficient early vision processing, the inclusion of processing circuitry within the pixels, prevents such systems from acquiring high-resolution images. These space constraints are eliminated if the processing is performed serially during read-out using pixel-block-parallel-addressing. Various kernels can be programmed in the processing unit of the imager and convolution is performed on readout with several kernels in parallel. Functionally, the image itself serves as an analog memory because the image dynamics occur at much slower speed than the image processing being performed. The benefits of this approach are: (1) small pixel size allowing for high-resolution imaging, (2) a single processor unit is used throughout the entire retina and (3) programmability does not impact the imaging array density. The space constraints are then transformed into temporal restrictions because the scanning clock speed and response time of the processing circuits must scale with the size of the array. This approach is often referred to as Computation-on-Readout (COR). This architecture is in fact very similar to the APS, differing in only that it includes a processing core; controlling the row/column selection and facilitating the COR.

Dallaire et al. [12] have applied the COR organisation by implementing a multiresolution edge detection algorithm on hexagonal pixel array. Another similar system based on COR was used by Mallik et al. [13], implementing a temporal change threshold detector. A more generic approach was taken by Gruev et al. [14, 10], attempting to realise a *pseudogeneral-purpose* vision chip for spatiotemporal filtering having the size and configuration of processing convolution kernels programmable.

3.4 Centroid Detection

Visual position tracking and centroid detection have been tasks traditionally associated with military applications. However these same tasks are fundamental to the more generic field of image recognition. Traditional image processing techniques effectively condition, filter and process image data but still normally output a matrix of pixels, constituting an

image. For perceptive vision applications it is paramount to cluster together pixels in a region of interest and provide a single entity for this. This task is often referred to as object segmentation. Having performed this, it is useful to describe the object using a centroid coordinate describing the object position and a single magnitude providing a measure for the object size. Having such high-level processed data available can benefit countless systems through a broad range of disciplines.

Machine vision for autonomous navigation, automation of security camera tasks, image stabilisation for medical applications and biochemical cellular migration/population analysis are some applications that could benefit from advances in such processing techniques. For mobile platforms including autonomous systems and handheld devices, minimising power consumption is of the upmost importance. It is therefore beneficial to include low-power front-end electronics to perform this saliency or region-of-interest detection to alleviate other processing tasks by applying attention only where it is most useful.

For applications that include the centroid computation as part of a feedback loop, high speed and low latency are most crucial. Latency is an especially important issue for image stabilisation and feedback systems with mechanical actuators. They demand quick response or risk becoming oscillatory or simply ineffective. For example, in microsurgery, tremors from the surgeons hand in addition to tremors from the subject can result in hazard situations. Here, optical tracking the surgeons instrument and objects in the operating field coupled with mechanical compensation of surgical instruments could result in jitter-free surgery.

3.4.1 Sequential Processing

As already stated, virtually all processing architectures today are based on the von Neumann sequential processing paradigm [15]. With the advent of reconfigurable processing hardware including field-programmable gate arrays (FPGA) and digital signal processors (DSP), it has been made possible to produce autonomous embedded systems. Furthermore, the reconfigurability has extended traditionally computer-bound software algorithms to be used in real-time processing hardware. The computational efficiency of a state-of-the-art hardware platform is more or less fixed; usually quoted in mW or μW per MIPS (Million Instructions Per Second). Subsequently, it is the software algorithm making good use of hardware resources which determines the system performance.

This subsection will outline some common software techniques suitable for object centroid computation.

Centre-of-Mass Computation

Calculating the centre of mass (COM) or centroid of an object is a relatively (computationally) simple and efficient task. Considering the image to be a matrix I of intensities that contains both an object and a background. Equation 3.1 gives the centroid calculation for a single axis.

$$C_x = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_i \cdot I_{ij})}{\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}} \tag{3.1}$$

where x_i is the coordinate of a pixel on the x-axis and I_{ij} is the intensity of that pixel. This equation assumes that the intensities of the object have higher numerical value than the background. The computational expense (related to image size) of a single axis centroid calculation is given by: $I_x \propto 2xy$, where I_x is the number of processor instructions and x and y are the image dimensions.

Although this approach is valid for asymmetric objects, the method is especially susceptible to changes in object shape and orientation between successive images when used in target tracking. Furthermore, this method alone cannot handle multiple target tracking/centroiding.

Object Segmentation

In order to facilitate Multi-Target Tracking (MTT), effective object segmentation becomes a fundamental requirement. A thresholding function to produce a binary template combined with an object indexing technique is a basic method for achieving this. However, in real-world images containing noisy undefined targets a more advanced technique is required for robust object segmentation.

The Active Contour (Snake) Algorithm [16]

The active contour method provides a reliable method of object selection within images containing "real-world" data. An active contour (snake) is a deformable contour that moves

under a variety of local image constraints and object-model constraints. The representation of a snake is: $v(s) = (x(s), y(s))$, where s runs from 0 to 1 over the perimeter of the snake. The snake is controlled by minimising a function which converts high-level contour information like curvature and discontinuities and low-level image information like edges, gradients and terminations into energies. The energy functional is given by:

$$E = \int_0^1 \frac{1}{2}[\alpha|x'(s)|^2 + \beta|x''(s)|^2] + E_{ext}(x(s))ds \tag{3.2}$$

where α and β are weighting parameters that control the snakes tension and rigidity, respectively and $x'(s)$ and $x''(s)$ denote the first and second derivatives of $x(s)$ with respect to s. The external energy function E_{ext} is derived from the image so that it takes on its smaller values at the features of interest, such as boundaries. Given a grey-level image $I(x, y)$, viewed as a function of continuous position variables (x, y), a typical external energy designed to lead an active contour toward a discrete edge is: $E_{ext}(x, y) = -|\nabla I(x, y)|^2$, where ∇ is the gradient operator. The energy function is minimised by solving the Euler equation:

$$\alpha x''(s) + \beta x''''(s) + \nabla E_{ext} = 0 \tag{3.3}$$

On formation of the active contour boundary, the area enclosed can be delegated to a centre-of-mass calculation function to compute the centroid of that particular object.

The active contour method provides an effective boundary detection technique suitable for object segmentation. The resulting contour is smooth and continuous, and adapts readily to deforming objects (changing shape, orientation and/or size). On the down side, the snake algorithm is computationally intensive and requires external initialisation processes which are able to position a snake close enough to the desired solution. Furthermore, the image data is required to be sufficiently smooth so that the snake does not remain blind to the desired solution and to maintain numerical stability in the iterative computations.

Other techniques

It has already been established that a software solution to multiple target tracking in real-world data needs to incorporate several algorithms. To perform object segmentation the

active contour method has been outlined, whereas for the centroid calculation the centre-of-mass computation has been mentioned. Other techniques include:

- ***For centroid calculation:*** object contour averaging, intensity centroiding [17], multithreshold centroiding [18] and Gaussian fit estimation [19].
- ***For object segmentation:*** binary thresholding, fuzzy [20], alpha map [21], particle filter [22], distribution matching [23] and gradient vector diffusion+region merging [24].

Limitations

Although sequential processing is extremely powerful in processing off-line data, this technique presents a computational bottleneck in real-time centroid processing; especially in the case of multiple-target tracking. The sheer volume of image data and complexity of object segmentation algorithms result in a computational workload that would render an embedded processing platform too power intensive for real-time and high frame-rate operation.

3.4.2 Distributed Processing

To overcome the computational limitations of traditional image processing techniques, a parallel processing approach could be explored. Hybrid focal-plane electronics combined with photodetection elements have the potential to achieve computationally efficient distributed centroid processing.

A review of previously developed vision chips demonstrate both the feasibility and potential in distributed architectures for centroid processing. High speed operation, good robustness and low power consumption have already been reported in a number of such systems already developed. A quantitative comparison of several centroiding vision chips in key performance criteria is provided in Table 3.2. The underlying principles of operation can be divided into three main categories:

Column/Row Summation

The most common approach to object centroid computation is illustrated in Fig. 3.5. The basic principle is to sum the pixels in all rows and columns to the edge of the array and

Year	Ref.	Tech (μm)	Die Size (mm^2)	Array Size	Pixel Side (λ)	Pixel Format	Fill Factor(%)	Multiple Objects	Object Sizing	Accuracy (Pixels)	Speed (KFPS)	Power (mW)
2005	[25]	0.18	25.00	48×48	944	Analogue, Binary[1]	12.5	yes	yes	1[3]	2	0.24
2004	[26]	0.18	4.83	64×64	179	Binary[3], Averaging	27	no	no	0.3	0.3	41
2004	[27]	0.5	-	43×43	58	APS[4]	49	yes[5]	no	0.2-0.42	-	-
2004	[28]	0.7	18.00	5×5	29	Analogue[34]	-	no	no	-	2.4-4.8	-
2004	[29]	0.8	6.25	20×20	36	Analogue[3]	12	no	no	0.013	3	15
2003	[30]	0.18	6.00	80×80	923	Binary[3]	51	no	no	0.1	1	30
2003	[31]	0.5	49.00	64×64	41	Binary[6]	18.8	18 max.[5]	no	1[3]	1	112
2003	[32]	0.6	20.25	11×11	17	Binary[6]	6.7	yes[5]	no	0.1	10	-
2003	[33]	0.8	8.00	20×20	36	Binary[3]	12	no	no	0.1	3	15
2002	[34]	0.5	2.25	24×24	133	Analogue[4]	19	no	no	-	1	-
2002	[35]	0.6	2.88	60×36	76	Analogue[3], APS[7]	12, 16[7]	no	no	1[3]	3.6[8]	2.6
1999	[36]	0.5	20.25	43×43	57	Binary[3]	23	no	no	-	-	-
1999	[37]	1.2	4.00	40×1	188	Analogue[9]	5	no	no	-	8[8]	-
1999	[38]	1.2	22.09	25×24	266	Analogue[9]	3	no	no	-	-	19.9
1999	[39]	2.0	4.93	23×1	298	Analogue[9]	-	no	no	-	-	5.4
1998	[40]	2.0	6.00	24×24	62	Analogue[9]	30	no	no	3	7[8]	0.25[10]
1998	[41]	1.2	4.00	12×10	183	Analogue[3]	11	no	no	1[3]	-	5
1998	[42]	3.0	-	25×1	-	Neuron[3]	-	no	no	1.5	-	10[11]

[1] Centroiding by distributed spatiotemporal algorithm; presented in this thesis.
[2] Sub-pixel accuracy possible by post-processing.
[3] Centroiding by column and row summation (centre-of-mass).
[4] Computation-on-readout (not in-pixel).
[5] Multiple centroiding by sequential windowing and scanning.
[6] Centroiding by thresholding, windowing and searching.
[7] Includes interweaved APS imager combined within centroid (and motion detecting) array.
[8] Tracking speed, i.e. stimulus moving at pixels/sec.
[9] Centroiding by Winner-Take-All (WTA) network.
[10] Only static power dissipation is reported.
[11] Excluding image acquisition, i.e. this system includes only centroiding hardware.

Table 3.2: Comparative review of centroid detecting vision chips.

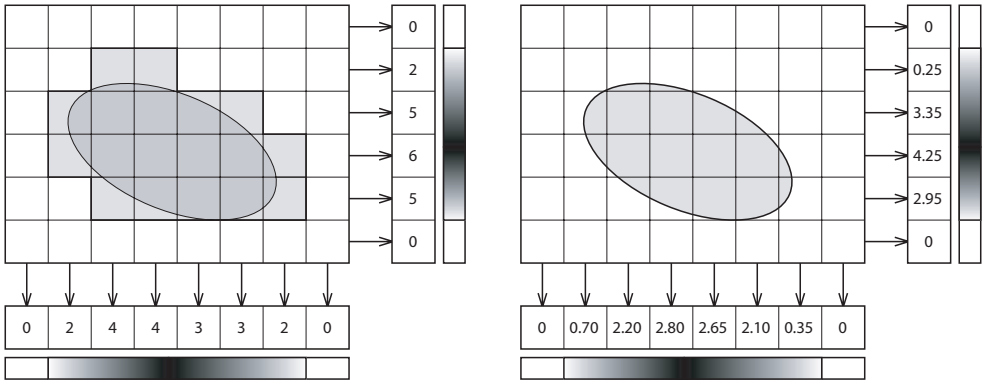


Figure 3.5: Object centroid computation by row/column summation followed by one dimension mean point calculation. Binary (left) and analogue (right) representation schemes.

use one-dimensional mean point calculation to determine the X and Y centre of mass. This mean point calculation can be implemented either using analogue techniques, for example using a basic vector-matrix multiplier or with a mean cumulative computation function implemented with digital logic.

The summation has been executed upon either binary values (employing a thresholding function) [26] [30] [33] [36], or directly on continuous value (analogue) data [28] [34] [35] [29] [24]. Since the analogue summation considers the fractional values of the object edge pixels, centroid computation with sub-pixel accuracy is possible using this technique. The discrete summation however provides am increased degree of immunity to noise and/or fuzzy data.

Although this column/row summation technique can compute an object centroid with precision, speed and relatively simple hardware, this technique remains limited to computing single object centroids.

Windowing and Search

Another effective technique, more widely used in general target tracking is using windowing and search operations to locate objects and then to locally compute the centroid. Systems based on such a strategy employ an initial search algorithm to locate different points of interest. This has been done by using maximum local point detectors on a resistive network and then generating a search window by propagating outwards [32] or by target tracking; using an initial target template (lock) to define the initial search windows [31]. On definition of these tracking windows, the centroid extraction is normally facilitated through the

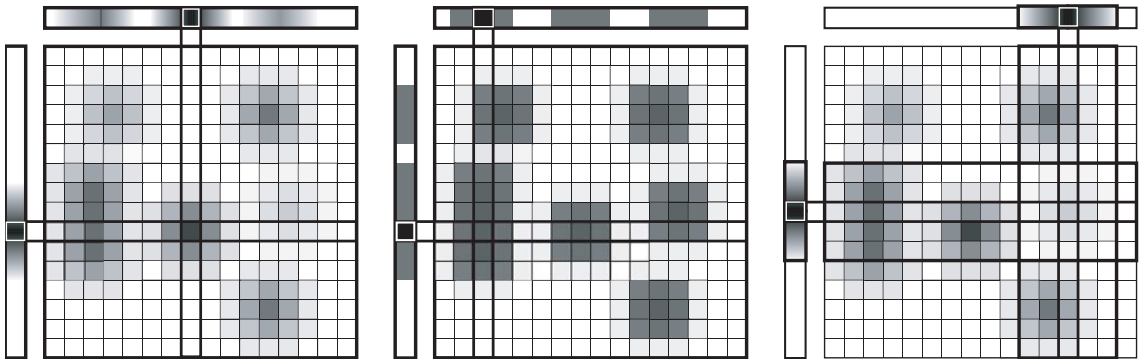


Figure 3.6: Object centroid computation by winner-take-all network (competition illustrated only in one dimension). Using basic resistive grid (left) for single centroid, dynamic switching network (middle) for saliency/object segmentation and interrogation window (right) for centroid tracking

column/row summation method.

The great advantage of such schemes being the ability to track and centroid multiple targets. However, this is at the expense of increased complexity, either with intricate in-pixel digital processing or off-chip programmable logic device (PLD) to facilitate the digital computation. Furthermore this introduces the requirement for a clock and control strategy which can lead to degradation of signal-to-noise ratio (SNR) in addition to increased power consumption.

Winner-take-all (WTA)

A third approach to centroid detection is to employ analogue processing for object segmentation and/or saliency followed by a winner-take-all network on a resistive grid. Such systems have either operated on the resistive network directly [40] or included dynamic switch networks [38] for object segmentation or spatial derivative circuits [39] [37] for direction sensing. These various techniques are illustrated in Fig. 3.6.

As this is a fundamentally analogue solution, reduced circuit complexity makes implementation straight forward. The voltage-smoothing resistive grid inherently removes noise and thus increases robustness. Furthermore, by combining WTA networks with windowing functions, multiple object centroid determination is possible [40].

3.5 Summary

This chapter has begun by reviewing CCD and CMOS technologies. It has been established that these technologies will continue to co-exist as complimentary rather than one superseding the other. CCD's will continue to provide high-quality, high resolution imagers for top-end applications, whereas CMOS offers the ability of in-pixel integrability with electronics and low power/high speed operation. This trend makes CMOS technology ideal for implementation of biologically inspired electronics (mentioned previously), based on distributed algorithms and architectures.

Conventional software techniques have extensively tackled single target tracking and centroid detection. The algorithm involved is both computationally efficient and hardware implementable. Moreover, many distributed techniques based on the same centre-of-gravity calculation have also proved successful in implementing robust, accurate and power efficient vision chips.

Centroiding vision chips however, have failed to deliver true distributed multi-target tracking. Although few systems have reported multiple object capability, the effectiveness and efficiency has yet to be demonstrated. On the other hand, software techniques have matured a variety of object segmentation algorithms suitable for multi-target tracking. However, the huge computational expense of implementing these limit their use to offline or low frame-rate processing. Furthermore, such techniques do not scale efficiently, for example, the array (pixel grid) size may only be scaled a certain amount; until the processing platform is operating at full capacity.

Another important feature vision chips have not yet yielded upon is object size and/or shape estimation. This coupled with centroid location would provide very powerful high-level processed data widely applicable in vision processing. For example, object size could be used as a screening parameter, limiting the centroid co-ordinates only to objects within a certain size window. This would be useful in microscopic cellular population analysis for counting of biological cells and classification by size. Such a technique could provide cell-type ratio's, for example, between red and white blood cells; that have a distinct size difference.

References

[1] W. S. Boyle and G. E. Smith, "Charge coupled semiconductor devices," *Bell System Technology Journal*, vol. 49, pp. 587–593, 1970.

[2] E. R. Fossum, "Active Pixel Sensors (APS) - Are CCDs Dinosaurs?," *Proceedings of SPIE*, vol. 1900, pp. 2–14, 1992.

[3] D. Litwiller, "CCD vs. CMOS: Facts and Fiction," *Photonics Spectra*, pp. 154–158, 2001.

[4] N. Blanc, "CCD versus CMOS has CCD imaging come to an end?," *Photogrammetric Week '01, by Fritsch/Spiller (eds.)*, pp. 131–137, 2001.

[5] L. Kozlowski, G. Rossi, L. Blanquart, R. Marchesini, Y. Huang, G. Chow and J. Richardson, "A Progressive 1920x1080 Imaging SoC for HDTV Cameras," *Proceedings of IEEE International Solid-state Circuits Conference*, vol. 19.7, 2005.

[6] M. Mase1, S. Kawahito1, M. Sasaki and Y.Wakamori, "A 19.5b DR CMOS Image Sensor with 12b Column-Parallel Cyclic A/D Converters," *Proceedings of IEEE International Solid-state Circuits Conference*, 2005.

[7] Safehouse Technology Limited Website, *http://www.safehouse.com.au*, 2004.

[8] C. A. Mead, "Neuromorphic Electronic Systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. pp1629–1636, 1990.

[9] P. Dudek and P. J. Hicks, "A General-Purpose Processor-per-Pixel Analog SIMD Vision Chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 1, pp. 13–52, 2005.

[10] V. Gruev and R. Etienne-Cummings, "A pipelined temporal difference imager," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 3, pp. 538–543, 2004.

[11] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro and S. E. Meana, "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Towards VSoCs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 851–863, 2004.

[12] S. Dallaire, M. Tremblay, and D. Poussart, "Mixed-signal VLSI architecture for real-time computer vision," *Real-Time Imaging*, vol. 3, no. 5, p. 307317, 1997.

[13] U. Mallik, M. Clapp, E. Choi, G. Cauwenberghs and R. Etienne-Cummings, "Temporal Change Threshold Detection Imager," *Proceedings of IEEE International Solid-state Circuits Conference*, vol. 19.9, pp. 23–25, 2005.

[14] V. Gruev and R. Etienne-Cummings, "Implementation of Steerable Spatiotemporal Image Filters on the Focal Plane," *IEEE Transactions on Circuits and Systems I: Analog and Digital Signal Processing*, vol. 49, no. 4, pp. 538–543, 2002.

[15] J. Von Neumann and A. W. Burks, *Theory of self-reproducing automata*. University of Illinois Press, 1966.

[16] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[17] M. Singh, B. S. Chauhan and N. K. Sharma, "VLSI Architecture of Centroid Tracking Algorithms for Video Tracker," *Proceedings of the 17th IEEE International Conference on VLSI Design*, pp. 697–700, 2004.

[18] J. Shah, "Applications and Implementations of Centroiding using CMOS Image Sensors," Master's thesis, University of Waterloo, Ontario, Canada, 2002.

[19] P. J. Pietraski and Z. Zojceski, "Search Fast Centroid Estimation Algorithm for High-Rate Detectors Based on a Two-Poinbt Gaussian Fit," *IEEE Transactions on Nuclear Science*, vol. 47, no. 4, pp. 1510–1515, 2000.

[20] B. M. Carvalho, G. T. Herman and Y. T. Kong, "Simultaneous Fuzzy Segmentation of Multiple Objects," *Electronic Notes in Discrete Mathematics, by A. Del Lungo, V. Di Ges and A. Kuba, eds.*, 2003.

[21] Y. Altunbasak, R. Oten and R. J. P. de Figueiredo, "Simultaneous object segmentation, multiple object tracking and alpha map generation," *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. 69–72, 1997.

[22] Z. Khan, T. Balch and F. Dellaert, "Efficient particle filter-based tracking of multiple interacting targets using an mrf-based motion model," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 254–259, 2003.

[23] D. Freedman, R. J. Radke, Y. Jeong, T. Zhang and G. T. Y. Chen, "Model-Based Multi-Object Segmentation via Distribution Matching," *Proceedings of the IEEE Workshop on Articulated and Nonrigid Motion*, p. 11, 2004.

[24] Z. Yu and C. Bajaj, "Image segmentation using gradient vector diffusion and region merging," *Proceedings of the IEEE International Conference on Pattern Recognition*, vol. 2, pp. 941–944, 2002.

[25] T. Constandinou, *Bio-inspired Electronics for Micropower Vision Processing.* PhD thesis, Imperial College of Science, Technology and Medicine, University of London, 2005.

[26] C. Thomas, R. Hornsey and K. Yip, "CMOS imager design for fast centroid readout," *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, vol. 4, pp. 2315–2318, 2004.

[27] A. Fish, D. Akselrod and O. Yadid-Pecht, "High Precision Image Centroid Computation via an Adaptive K-Winner-Take-all Circuit in Conjunction with a Dynamic Element Matching Algorithm for Star Tracking Applications," *Kluwer Analog Integrated Circuits and Signal Processing*, vol. 39, pp. 251–256, 2004.

[28] B. H. Pio, B. Hayes-Gill, M. Clark, M. G. Somekh, C. W. See, S. Morgan and A. Ng, "Integration of a Photodiode Array and Centroid Processing on a single CMOS Chip for a Real-time Shack-Hartmann Wavefront Sensor," *IEEE Sensors Journal*, vol. 4, no. 6, pp. 787–794, 2004.

[29] N. Massari, L. Gonzo, M. Gottardi and A. Simoni, "A Fast CMOS Optical Position Sensor with High subpixel Resolution," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 1, pp. 116–123, 2004.

[30] R. D. Burns, J. Shah, C. Hong, S. Pepic, J. S. Lee, R. I. Hornsey and P. Thomas, "Object Location and Centroiding Techniques with CMOS Active Pixel Sensors," *IEEE Transactions on Electron Devices*, vol. 50, no. 12, pp. 2369–2377, 2003.

[31] T. Komuro, I. Ishii, M. Ishikawa and A. Yoshida, "A Digital Vision Chip Specialized for High-Speed Target Tracking," *IEEE Transactions on Electron Devices*, vol. 50, no. 1, pp. 191–199, 2003.

[32] J. Akita, A. Watanabe, O. Tooyama, M. Miyama, M. Yoshimoto, "An Image Sensor with Fast Objects' Position Extraction Function," *IEEE Transactions on Electron Devices*, vol. 50, no. 1, pp. 184–190, 2003.

[33] N. Viarani, N. Massari, L. Gonzo, M. Gottardi, D. Stoppa and A. Simoni, "A fast and low power CMOS sensor for optical tracking," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 796–799, 2003.

[34] R. A. Blum, C. S. Wilson, P. E. Hasler and S. P. DeWeerth, "A CMOS imager with real-time frame differencing and centroid computation," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 329–332, 2002.

[35] M. A. Clapp and R. Etienne-Cummings, "A Dual Pixel-type Array for Imaging and Motion Centroin Localization," *IEEE Sensors Journal*, vol. 2, no. 6, pp. 529–548, 2002.

[36] G. Erten and S. Hagopian, "Integrated image sensor processor with on-chip centroiding," *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, vol. 1, pp. 262–265, 1999.

[37] G. Indiveri, "Neuromorphic analog VLSI sensor for visual tracking: circuits and application examples," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 11, pp. 1337–1347, 1999.

[38] C. S. Wilson, T. G. Morris and S. P. DeWeerth, "A two-dimensional, object-based analog VLSI visual attention system," *Proceedings of the 20th Conference on Advanced Research in VLSI*, pp. 291–308, 1999.

[39] T. Horiuchi and E. Niebur, "Conjunction search using a 1-D, analog VLSI-based, attentional search/tracking chip," *Proceedings of the 20th Conference on Advanced Research in VLSI*, pp. 276–290, 1998.

[40] V. Brajovic and T. Kanade, "Computational sensor for visual tracking with attention," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 8, pp. 1199–1207, 1998.

[41] R. Etienne-Cummings, V. Gruev and M. Abdel-Ghani, "VLSI Implementation of Motion Centroid Localization for Autonomous Navigation," *Advances in Neural Information Processing Systems*, vol. 10, pp. 685–691, 1998.

[42] N. M. Yu, T. Shibata and T. Ohmi, "A Real-Time Center-of-Mass Tracker Circuit Implemented by Neuron MOS Technology," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 4, pp. 495–503, 1998.

Chapter 4

A Distributed Algorithm for Centroid Detection

4.1 Introduction

This chapter introduces a novel scheme suitable for object-based centroid computation based on a distributed processing architecture. Although several of the features have been biologically inspired, the algorithm is fundamentally synthetic. By using this hybrid approach, a realistically hardware implementable system can be developed benefiting from increased computational efficiency provided by the bio-inspired analogue processing elements. The reduced power consumption enables realisation of mobile diagnosis devices which would otherwise be technically unachievable.

This chapter begins by describing qualitatively a specific distributed algorithm suitable for hardware implementation. A formalisation is then provided by describing the various parallel processing functions using mathematical and/or logical expressions. It is then outlined how this would be implemented in hardware, from a top-level functional perspective. Subsequently, software simulation results provide details to computational workload, robustness and accuracy. Finally, this algorithm is architecturally generalised to a parallel-processing platform with other similarly-based distributed algorithms being proposed.

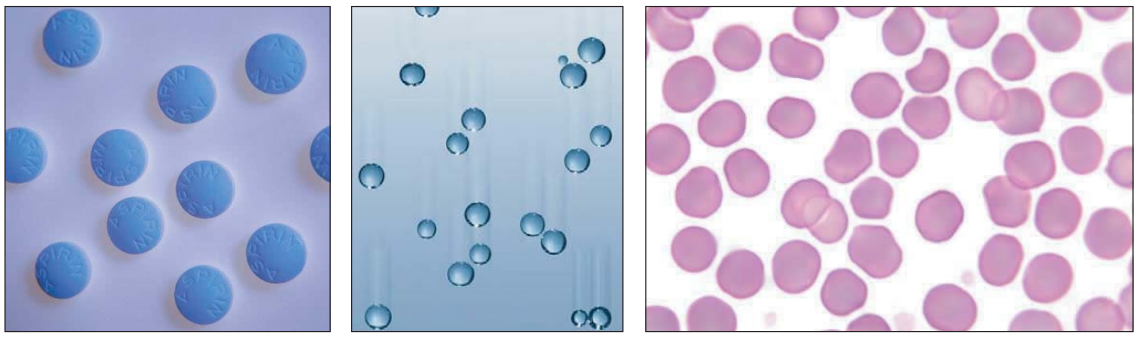


Figure 4.1: Example analysis scenarios where extraction of object centroid and/or size could provide useful information in (from left to right): (a) Pharmaceutical drug production (b) Reliability (leak) detection of bubbles in fluids and (c) Microscopic cellular population analysis

4.2 The Bio-pulsating Contour Reduction Algorithm

An algorithm is proposed for distributed centroid processing and sizing of simple objects [1]. Circular *blob*-like objects with uniform texture and an intensity differing from the background level can be segmented and their size and position (centroid) determined by means of a distributed binary algorithm. Some example images this can be applied to are shown in Fig. 4.1.

Possible applications for such perceptive vision processing span through many disciplines. From machine vision applications such as production line inspection and reliability detection, security (asset tracking), surveillance (counting persons passing a boundary), space (star tracking, lunar mapping and vehicle navigation) and military (target tracking) to biomedical analysis (cellular, microbial and neural).

4.2.1 Overview

This algorithm uses an edge-detection technique to form the contours and trigger the data-driven processing. On detection of an object boundary, the initial state for the signal flow is set. By propagating an inward fill, the contour can be reduced until this converges to the centre. The central point is detected by utilising spatiotemporal integration; i.e. a summation of the cells set within the receptive field within a certain time window. On centroid detection, the object is reset and output transmitted, thus realising an inward pulsating action. The frequency of pulsation determines the size, i.e. radius of this object.

Fig. 4.2 illustrates this interaction graphically through computer simulations (described later in detail).

4.2.2 Method

Threshold Detection

Objects are defined as regions in the image with narrow-field (2x2 average with adjacent cells[1]) local-average intensity either below (or above) the average level of the input image. This can implemented either as a true global average or a wide-field local-average; centred on the object pixel to be tested.

Edge and Contour Detection

The edges are detected (in continuous-time) by computing the difference in adjacent cellular intensities. In a quad-grid connectivity scheme (i.e. using square cells), each pixel is compared with its four neighbours. Subsequently contours are formed if a continuous edge is determined; assuming that a contour is defined as a cell corner adjacent to at least two detected edges.

State Setting

The pulsating action is initially triggered by setting a cells (binary) state on detection of a contour, i.e. when it lies on a continuous edge. The reduction is then facilitating by each pixel checking whether any of their neighbouring cells have been set in addition to the object criterion (threshold for that cell) being satisfied. The rate of this contour (cellular state) reduction is defined by an artificial propagation delay introduced in this event path.

Centroid Detection

In parallel with this contour reduction, each pixel checks whether it satisfies the centre-criterion. This is defined as when surrounding cells (but not directly adjacent; for reliability)

[1] A cell is referred to as the minimally-repetitive processing element (pixel circuitry) in for hardware tessellation.

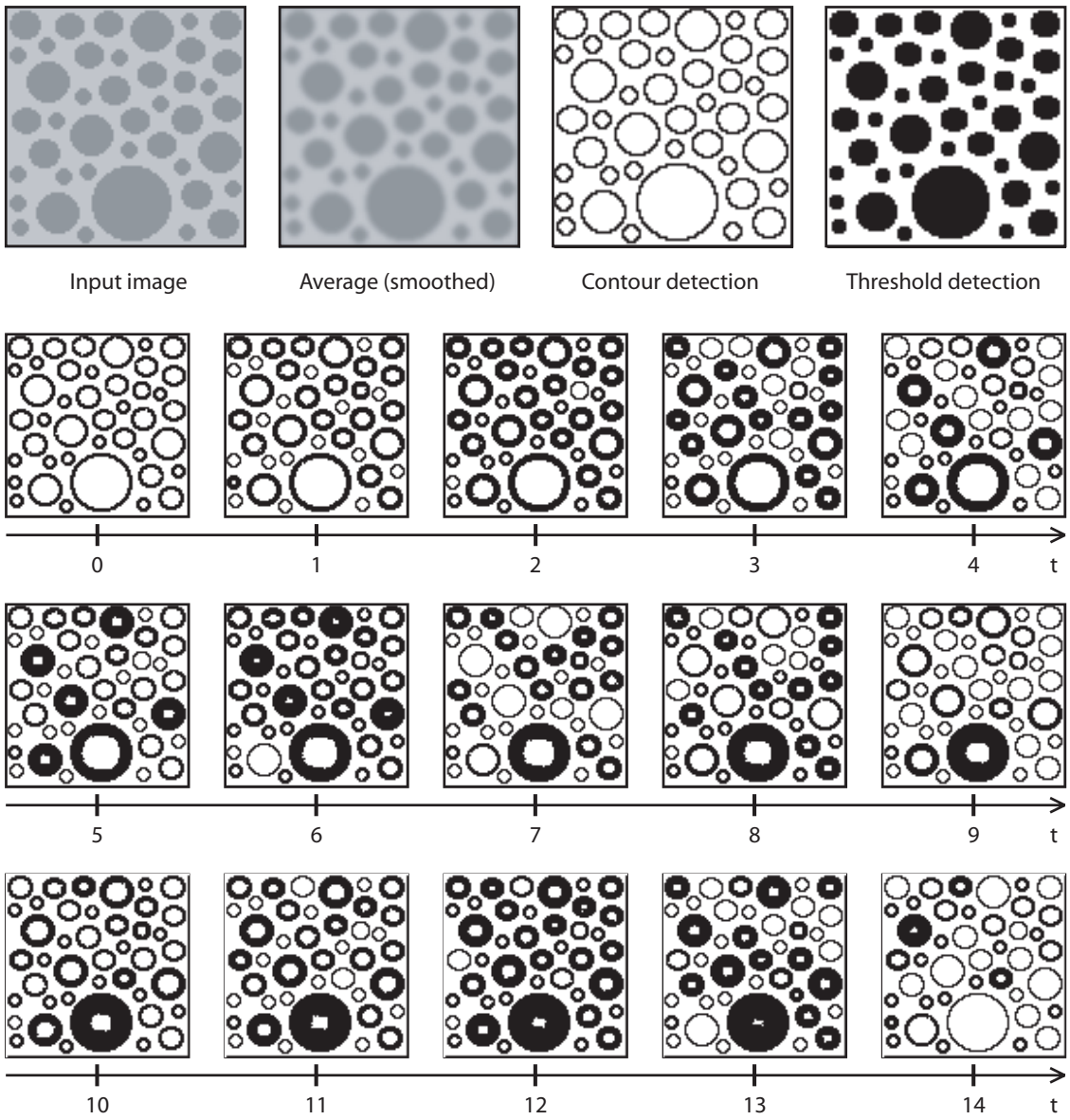


Figure 4.2: Computer simulation results of the bio-pulsating contour reduction algorithm, illustrating continuous-time image processing functions (top row) and snapshots taken at regular time intervals at the propagation delay of the processing (bottom 3 rows)

are set and the central pixel is remains unset. Such a condition immediately flags a centroid-detection signal, that transmits the pixel co-ordinates off-array and subsequently off-chip, then issuing a localised reset signal.

State Resetting

This reset signal is then back-propagated outwards in a recursive manner similar to the contour reduction with the absence of the artificial delay. This delay is not required in both paths, as the pulsating (or reset) period can be defined by including a delay in either the forward or back-propagating path. More importantly a swift back-propagating reset is required to avoid flagging multiple centroids. The resetting action therefore acts by suppressing neighbouring cells from detecting centroids, thus realising a winner-takes-all (WTA) type functionality.

An unusual and important feature of this method is the absence of any pre-defined synchronisation signal, for example, a clock. The only synchronisation is obtained through the data-driven object reset scheme but on a local, rather than a global basis. This in combination with the artificial delay time-constant defines the processing time, since CMOS the (asynchronous) digital logic operates with propagation delays in the order of nanoseconds.

4.2.3 Analytical Formalisation

Image Processing Functions

The distributed spike-domain processing is driven by two specific binary signals generated within each cell; the THRESHOLD (Th) and CONTOUR (Co) inputs. These serve to initiate and steer the signal propagation correctly.

The THRESHOLD input is used to facilitate the object segmentation by defining the valid area the signal can propagate within. Therefore this has the task of ensuring the fill is propagating *inwards* and NOT outwards. This input for a particular cell is generated by comparing its intensity, with the average background (outside object boundaries) intensity. For object and background intensities that significantly differ it is therefore valid to use the average image intensity as the global threshold point.

As the centroid processing occurs at the pixel corners, a local average of all pixels adjacent to that processing node is required to convey a valid intensity at that point. This

has the additional advantage of smoothing the image, thus helping reduce noise. This local averaging (narrow-field) function can be expressed as given in Eqn. 4.1. Further noise suppression can be achieved by implementing a larger averaging field, eg. using a 9 or 16 pixel squares.

$$Av_{narrow} = \frac{1}{4}\sum_{i=x}^{x+1}\sum_{j=y}^{y+1} I_{i,j} \tag{4.1}$$

Similarly, to calculate the global average intensity the summations extend though the whole array, as expressed in Eqn. 4.2.

$$Av_{global} = \frac{1}{xy}\sum_{i=1}^{x}\sum_{j=1}^{y} I_{i,j} \tag{4.2}$$

Where: (x,y) are the array dimensions.

However, for images with varying background intensity, for example a gradient due to lighting conditions, it is favourable to perform a *wide-field* local average. This should be a large enough area to extend beyond a single object in order to capture the background intensity. An easily hardware-implementable scheme would be to combine all the the narrow-field local averages calculated in a cells column and/or row to calculate the wide-field local average. This would provide a "global" average unique to every pixel as expressed in Eqn. 4.3.

$$Av_{wide} = \frac{1}{2x}\sum_{i=1}^{x} I_{narrow(i,j)} + \frac{1}{2y}\sum_{j=1}^{x} I_{narrow(i,j)} \tag{4.3}$$

Comparing the wide-field average to the narrow-field average constitutes the required thresholding function, given in Eqn.4.4.

$$Th = vdd \cdot H(Av_{narrow(x,y)} - Av_{wide(x,y)} + K) \tag{4.4}$$

Where: vdd is the supply voltage, $H(x)$ is the Heaviside function, $Av_{narrow(x,y)}$ and $Av_{wide(x,y))}$ are the narrow- and wide- field local averages centred on the (x, y) pixel and K is a tolerance adjustment constant; to avoid erroneous thresholding due to noise in images with low contrast selectivity.

To increase versatility to input images, adapting the threshold function such that the narrow- and wide- field averages can be inter-changed, would enable the system to select objects of either higher or lower intensity (relative to background).

In addition to requiring the THRESHOLD (Th) input to define the object areas, a CONTOUR (Co) input is also necessary to initiate the signalling from object boundaries. This can be generated by using an edge detection function followed by some basic post-processing to ensure reliability. To facilitate the edge detection, every pixel must be compared to its four adjacent neighbours and if the differential surpasses a certain threshold, an edge has been detected. On a tessellating basis, each pixel needs to be compared with only two neighbours, for convenience the pixels below and to the right. The corresponding expressions are given in Eqns. 4.5 and 4.6.

$$E_{vertical} = vdd \cdot H(|\ln(I_{x,y}) - \ln(I_{x,y+1})| - E_{threshold}) \tag{4.5}$$

$$E_{horizontal} = vdd \cdot H(|\ln(I_{x,y}) - \ln(I_{x+1,y})| - E_{threshold}) \tag{4.6}$$

Where: vdd is the supply voltage, $H()$ is the Heaviside function, $I_{x,y}$ is the intensity of the (x, y) pixel in the array and $E_{threshold}$ is the pre-defined threshold, for flagging an edge condition. Rather than comparing the intensity values directly, their natural logarithms are taken, for this achieves increased dynamic range for edge detection.

On determining how many edges surround a specific cell, a contour is defined as a continuous edge, i.e. when a cell is adjacent to two edges and lies on the outside of an object. This can be defined as a boolean expression, given in Eqn. 4.7.

$$Co = \overline{(Th)} \cdot \overline{(\overline{A} \cdot \overline{B} \cdot \overline{C}) + (\overline{A} \cdot \overline{B} \cdot \overline{D}) + (\overline{A} \cdot \overline{C} \cdot \overline{D}) + (\overline{B} \cdot \overline{C} \cdot \overline{D})} \tag{4.7}$$

Where: A, B, C and D are the four edge inputs and Th is the threshold status.

Centroid Detection Functions

Having used the continuous-time analogue processing (See Fig. 4.3) to generate two binary signals: CONTOUR and THRESHOLD, in turn these are used to feed, control and regulate the asynchronous logical *neuronal* network.

Each cell requires three bits static memory to store its current STATE, RESET and CENTRE activity. These are updated asynchronously depending on one cells CONTOUR and THRESHOLD inputs, its current STATE, RESET and CENTRE values and current STATE, RESET and CENTRE values from surrounding cells. The minimum required cellular interconnectivity for implementing this algorithm is illustrated in Fig. 4.4. This particular connectivity is required for the following reasons:

- Each pixel receives four state inputs (from directly adjacent cells) to facilitate the inward signal propagation.
- Each pixel receives four reset inputs (from directly adjacent cells) to facilitate the back-propagation signal, i.e. the local reset.
- Each pixel receives eight centre inputs (from directly and diagonally adjacent cells) to ensure multiple centroids are not detected. This realises a form of inhibition on centroid detection; preventing neighbouring cells from also registering a centre.
- Each pixel receives four state inputs (from indirectly adjacent cells, i.e. two pixels apart in each direction) to determine the surround status used in centroid determination. The immediately adjacent cells are no used for this purpose as this could pose a reliability issue for uneven aspect ratio objects, i.e. those that are not perfectly circular.

Therefore, the internal functionality of a (binary) processing element (cell) can be described using state diagrams or boolean expressions. These can be used to define the conditions for *setting* and *resetting* the various memories (RS flip-flop based).

The Set STATE (S_{Set}) condition is defined in Eqn. 4.8.

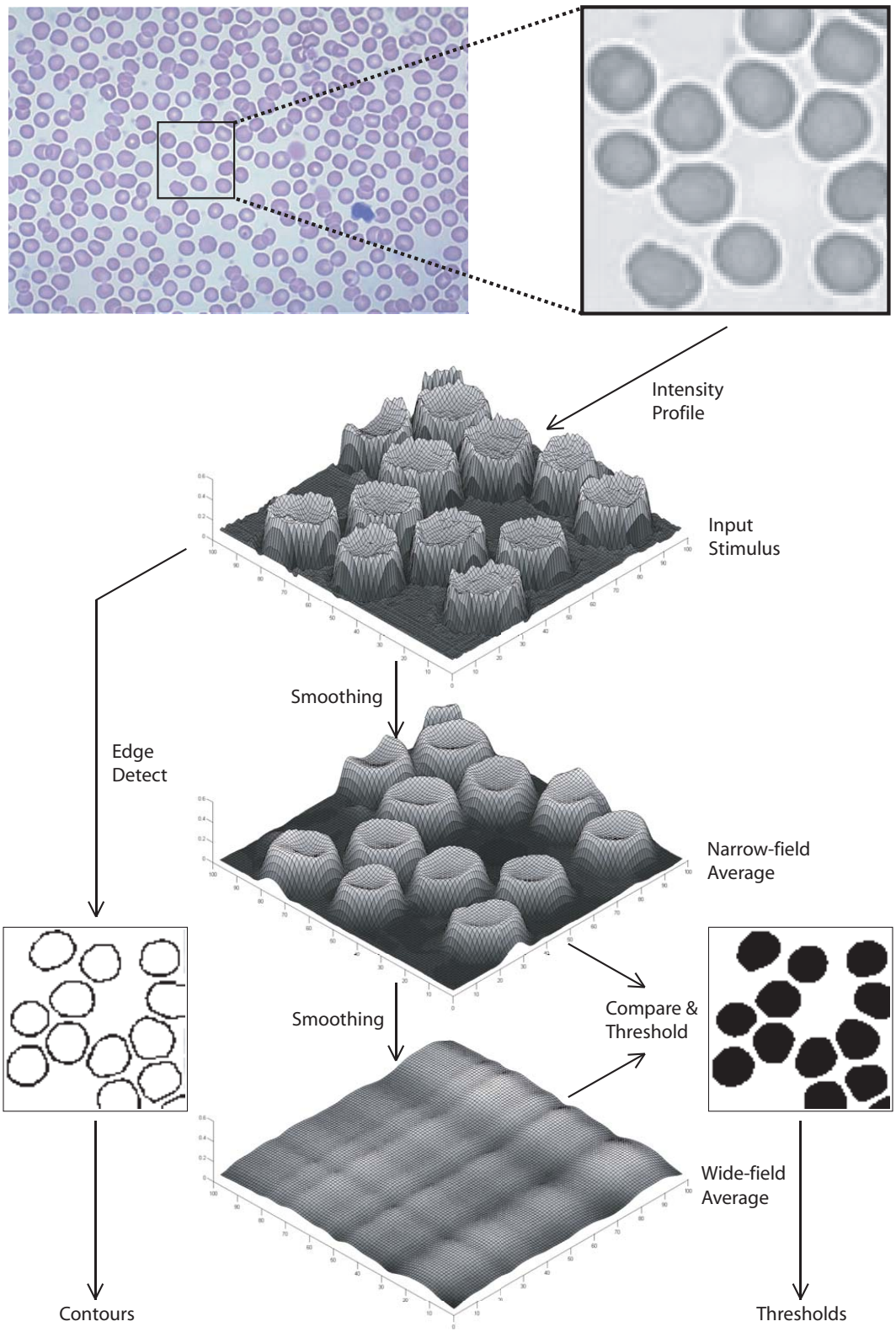


Figure 4.3: Front-end continuous-time image-processing functionality.

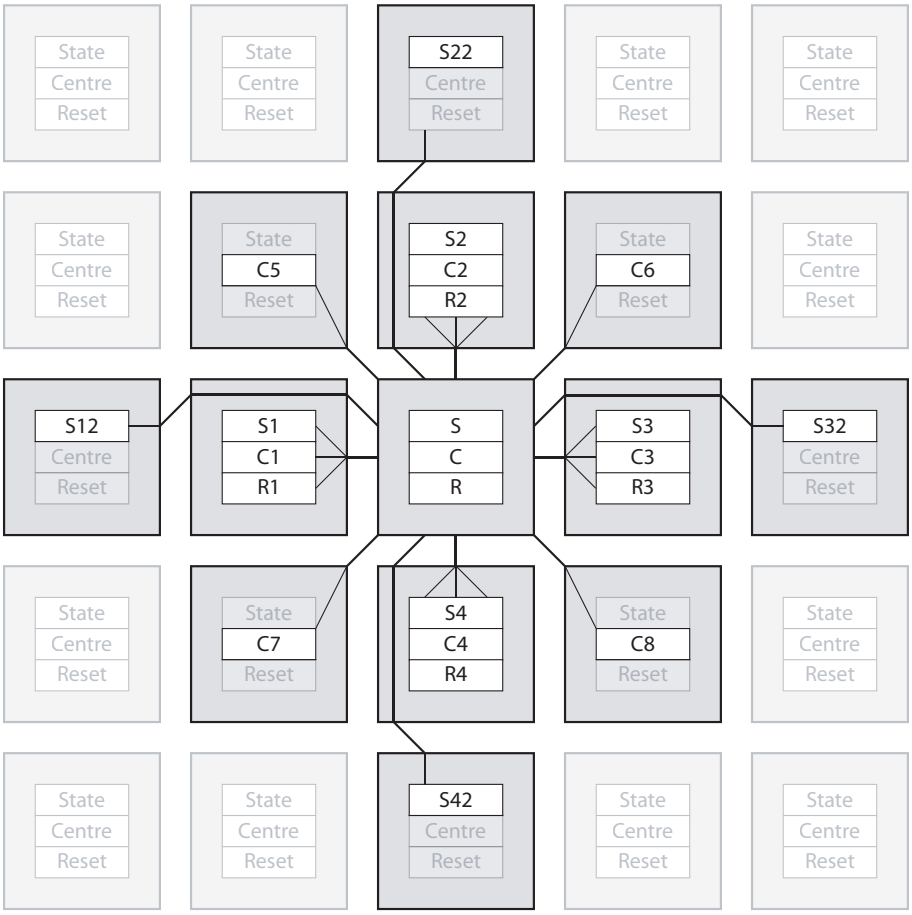


Figure 4.4: Local connectivity required for binary signals (state, reset and centre) to and from every pixel.

$$S_{Set}(t+\tau_s) = Co(t) + Th(t) \cdot (S_1(t) + S_2(t) + S_3(t) + S_4(t)) \tag{4.8}$$

Where: τ_s is the delay time-constant which defines the propagation rate, $Co(t)$ is the current CONTOUR status, $Th(t)$ is the current THRESHOLD status and $S_1(t)$, $S_2(t)$, $S_3(t)$, $S_4(t)$ being the STATE variables of the directly adjacent cells.

The Reset STATE (S_{Reset}) and Set RESET (R_{Set}) occur for identical conditions, i.e. when a cells RESET memory is set, its STATE memory resets. This occurs either on CENTRE being reset, or on RESET back-propagating (from adjacent cells) during a local reset. This condition is defined in Eqn. 4.9.

$$R_{Set}(t+\delta t) = S_{Reset}(t+\delta t)$$

$$S_{Reset}(t+\delta t) = C(t-\delta t)\cdot\overline{C}(t) + S(t)\cdot(R_1(t)+R_2(t)+R_3(t)+R_4(t)) \tag{4.9}$$

Where: δt represents the propagation delay of the combinational logic, $S(t)$ is the cells current STATE value, $C(t)$ is the current CENTRE status and $R_1(t)$, $R_2(t)$, $R_3(t)$, $R_4(t)$ are the RESET variables of the directly adjacent cells.

The RESET memory is configured to self-reset, i.e. operate as a monostable, by feeding the current RESET status through a small delay to the reset input. This ensures all setup and hold times are respected when issuing a back-propagating RESET signal. This condition is defined in Eqn. 4.10.

$$R_{Reset}(t+\tau_r) = R_{Set}(t) \tag{4.10}$$

Where: τ_r is the Set-to-Reset delay time-constant.

The CENTRE memory is set on detection of an OFF-centre, ON-surround condition, i.e. when surrounding pixels have STATE set and the centre-pixel has STATE not set. Furthermore, lateral inhibition (Eqn. 4.11) prevents a cell flagging CENTRE if an adjacent cells CENTRE status is set. This condition is defined in Eqn. 4.12.

$$C_{Inhibit}(t+\delta t) = C1(t)+C2(t)+C3(t)+C4(t)+C5(t)+C6(t)+C7(t)+C8(t) \tag{4.11}$$

Where: C1(t) to C8(t) are the CENTRE status of the 4 directly- and 4 diagonally- adjacent cells.

$$C_{Set}(t+\delta t) = \overline{C_{Inhibit}}\cdot\overline{S(t)}\cdot S_{12}(t)\cdot S_{22}(t)\cdot S_{32}(t)\cdot S_{42}(t) \tag{4.12}$$

Where: $S_{12}(t)$, $S_{22}(t)$, $S_{32}(t)$ and $S_{42}(t)$ are the STATE values of cells' two to the left, two above, two to the right and two below respectively.

Finally the CENTRE memory is reset when the contour reduction reaches the centre cell. This condition is defined in Eqn. 4.13.

$$C_{Reset}(t + \delta t) = \overline{S(t)} \cdot S_1(t) \cdot S_2(t) \cdot S_3(t) \cdot S_4(t) \tag{4.13}$$

Where: $S_1(t)$, $S_2(t)$, $S_3(t)$ and $S_4(t)$ are the STATE values of the directly adjacent cells.

4.2.4 Algorithmic Features

- Asynchronous: The distributed nature of this algorithm makes it readily implementable using asynchronous digital logic. A tuneable (artificial) propagation delay included in the input-enclosed feedback scheme provides a means of regulating and controlling the rate of pulsing, when operated in closed-loop mode. Furthermore such architectures are directly compatible with asynchronous off-chip communication protocols such as the address-event representation (AER) scheme.

- Parallel Processing: Being of distributed nature, each cell contains identical processing elements of relatively low specifications. This massive parallelism will be shown (later in Chapter 6) to result in exceptional system performance; including high speed and computational efficiency.

- Real-time: The parallel, distributed nature of the algorithm results in a very fast processing time, making real-time and very high frame-rate[2] processing feasible.

- Event-driven (spike domain): Image-detail driven processing produces a computational workload dependance on image content. This means that for no objects (in the focal-plane) to centroid, the computational burden is much reduced.

- Scalable: Identical processing elements working in parallel means scaling up to an increased array size will not produce a computational bottleneck. Power consumption is therefore directly proportional to the number of processing elements (pixels).

- Robustness: Parallel processing provides an inherent tolerance to various non-idealities. For example, effects of fabrication defects, process variations and ill-conditioned are reduced through relative representation and processing redundancy (discussed later).

- Hardware implementable: This distributed algorithm truly lends itself to hardware implementation in standard CMOS technologies. Combining weak inversion analogue

[2]Continuous-time asynchronous operation does not sample or refresh the image array at regular time intervals as in the case of a conventional imaging chip.

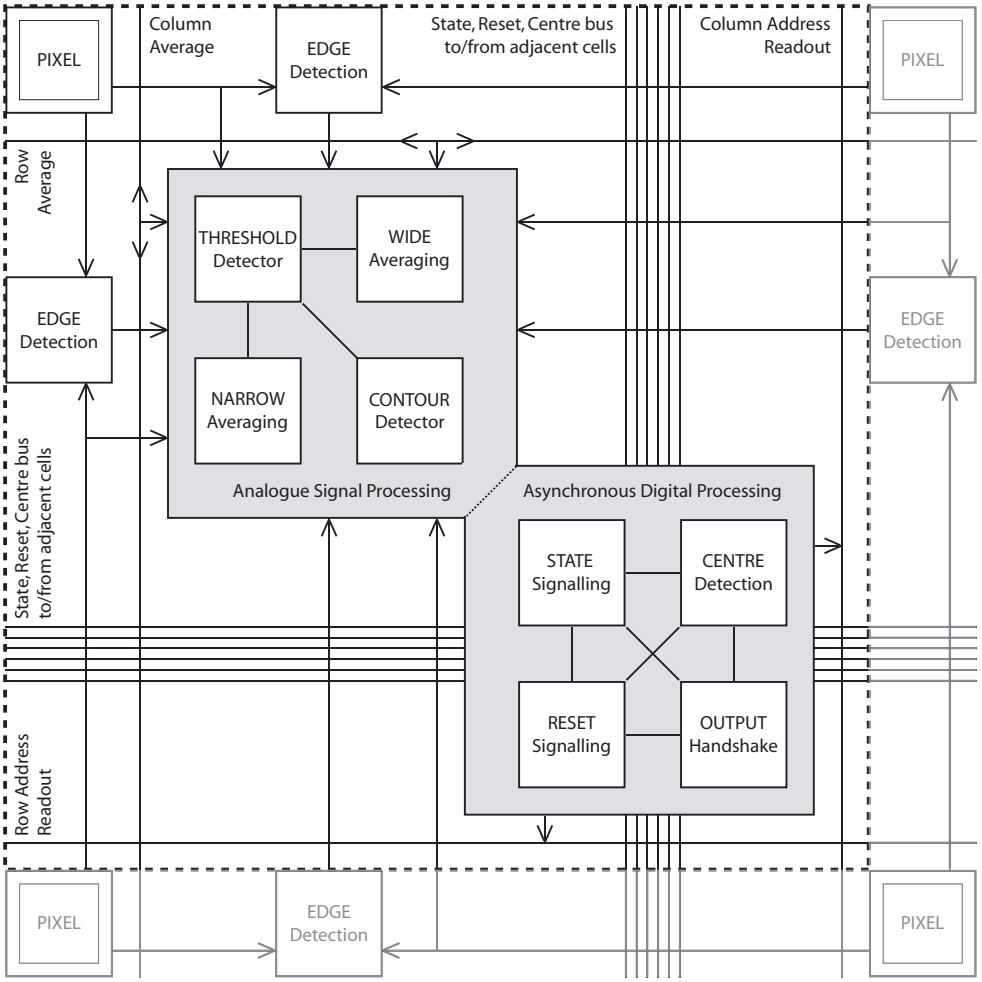


Figure 4.5: Proposed cellular architecture for object-based processing illustrating organisation and connectivity of functional blocks within a quad-pixel arrangement.

front-end signal conditioning and pre-processing together with distributed asynchronous digital for neuronal-like networks makes ultra low power CMOS implementation plausible.

4.2.5 Implementation

In order to facilitate the contour computation, the processing must occur at the pixel corners, as illustrated in the pixel-cell architecture shown in Fig. 4.5.

The various functional (cellular-level) blocks; all continuous-time topologies, to be im-

plemented in a standard CMOS technology are given below:

- Light detection: Integrated silicon pn junction (CMOS) photodiode (see Chapter. 5 for further details).
- Contour detection: Differential-input, single (discrete) output (thresholding) edge-detectors feeding combinational logic for contour detection.
- Threshold detection: Narrow-field averaging for input image smoothing and wide-field (or global) averaging for object detection; using current-mode techniques for linear computations.
- Local resetting: reconfigurable (dynamic) switch network (logical), regulated by threshold detector for object segmentation, to provide localised (object-constrained) resetting by back-propagation.
- Neuromorphic logic: performing delay-and-propagate computation for signal flow and centre-surround-like computation for centroid determination.
- Memory: Single-bit static memories for storage of each cells signal, reset and centroid states (3 bits per cell).

4.2.6 Simulation

Being an algorithm of both distributed and asynchronous nature, this is somewhat complex to model and virtually impossible to simulate the exact behaviour using conventional software techniques. However, by making some simplifying assumptions a "frame-based" representation can be derived and simulated.

The basic assumption is that all delay elements are perfectly matched and therefore all pixels in a frame can be exhaustively processed sequentially, for each delay "period". The sequence of this pixel processing can be implemented either in a scan fashion or for more realistic functionality, in a random pattern. This was developed using Borland Delphi V4 and the GUI developed is shown in Fig. 4.6. A full listing of the source code is provided in Appendix A.

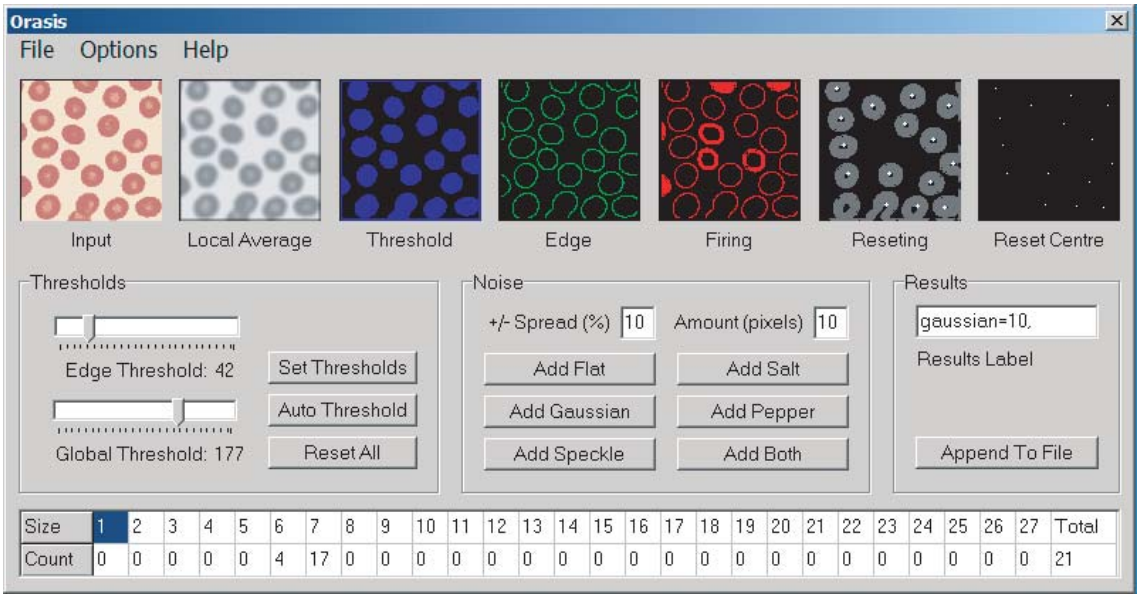


Figure 4.6: The ORASIS simulator: screenshot of the developed software simulator for the bio-pulsating contour reduction algorithm.

Effective Computation

In order to estimate the effective computation of this distributed algorithm, the frame-based equivalent algorithm needs to be evaluated. Although, many of the functions are static, i.e. require be executed each frame for each pixel, there also exist dynamic functions, for example the reset cycle being recursive. The static computation is dependant only on the array dimensions whereas dynamic computation is largely dependant on input data.

For a pixel array of dimensions (x, y), with n-objects $(a \in [0, n])$ of radius r_n-pixels each, the computational load required in processing each frame for main functions is specified in Table. 4.1. This frame-based algorithm could be optimised by combining some of the nested loop functions, thus reducing the computational workload by approximately 5-10%. On the other hand, memory access operations have not been included in determining the computational workload of the algorithm and as these are extensive, it is estimated these would increase the computational burden by at least 50-100% [2] [3] [4].

Having a frame capture and process time of $t\mu s$, the complete computational load becomes:

$$Computation(total) = \frac{1 + 8y + 37xy + 6\sum_{a=1}^{n}(\pi r_a^2)}{t \times 10^{-6}} \tag{4.14}$$

Function	Instructions[1]	Computational Load
Local Averaging	loops,[2] 4 ADD, 1 DIV	$y(1+5x)$
Global Averaging	loops,[2] 1 ADD, 1 DIV	$1+y(1+2x)$
Edge Detection	loops,[2] 4 SUB, 4 COND	$y(1+9x)$
Contour Detection	loops,[2] 4 COND	$y(1+5x)$
Threshold Detection	loops,[2] 1 COND	$y(1+2x)$
State Definition	loops,[2] 6 COND	$y(1+7x)$
Centre Definition	loops,[2] 5 COND	$y(1+5x)$
Reset Definition	loops,[2] 1 COND and per iteration,[3] 5 COND	$y(1+2x)+6\sum_{a=1}^{n}(\pi r_a^2)$

[1] INC = increment, COND = condition, ADD = addition / summation, SUB = subtraction, DIV = division.
[2] Double nested loops require y(1+x) INC/COND instructions.
[3] Multiple iterations due to recursive operation.

Table 4.1: Split of computational load for various processing functions

Estimating that the image comprises of n-objects of average radius r_{av}-pixels.

$$Computation(total) \simeq \frac{y(37x+8)+6n(\pi r_{av}^2)}{t\times 10^{-6}} \tag{4.15}$$

For computer simulation parameters, with 100x100 array @ 50fps with 25 objects of average 8-pixel radius:

$Computation(software) = 50(100(37(100)+8)+6(25)(\pi 8^2)) = 20.05MIPS$

For hardware operation parameters (see Chapter 6 for details), with 48x48 array @ 2000fps with 5 objects of average 5-pixel radius:

$Computation(hardware) = 2000(48*(37(48)+8)+6(5)(\pi 5^2)) = 175.21MIPS$

The advantage of implementing this algorithm in custom hardware using asynchronous techniques, is that more functions can be made dynamic and therefore data-driven. This results in a reduced *average* computational burden and can therefore contribute to substantial reductions in power consumption.

In contrast, a state-of-the-art digital signal processor (DSP), for example the Texas In-

struments TMS320C5000 series is quoted [5] to consume 0.25mW/MIPS. To execute the above mentioned (hardware) scenario on such a system (excluding imager consumption) would therefore consume at very least: 175.21*250μW=43.8mW. Furthermore, the maximum capacity of such power-efficient platforms are limited to 400-600MIPS. Therefore scaling this algorithm to a larger array size would be at the expense of frame-rate, thus making the realization of a high resolution, high-frame rate system unfeasible using existing DSP technology.

4.2.7 Robustness

The *robustness* of an algorithm provides an indication to its reliability against hardware fabrication non-idealities in addition to how immune it is to ill-conditioned or noisy data. Processing non-idealities refers to hardware fabrication defects (eg. unreliable via connections) in addition to process variations and device mismatches. In terms of this algorithm, process variations would effect the following:

- Component mismatch: This is by far the most critical expected source of errors. Component mismatch would directly effect the photodiode array, causing non-uniformities in offset (referred to as *fixed pattern noise*) and sensitivity (referred to as *speckle noise*) variations. Furthermore, non-uniformities in gain elements (transistors) would increase the degradation due to sensitivity mismatch. Beyond the imaging, edge detection and thresholding electronics would also be affected by component mismatches, resulting in uneven feature extraction.
- Processing Defects: Fabrication defects causing unreliable connections, could result in various signals being incorrectly conveyed from one cell to its neighbour. Such errors could cause a cell to flag an erroneous result or cause a break in the propagation cycle.

In order to analyse and evaluate the robustness of this algorithm, the input image data can be adjusted to include array non-uniformities such as fixed-pattern noise, pixel sensitivity and feature detection fluctuations. This can be modelled and/or simulated using various techniques.

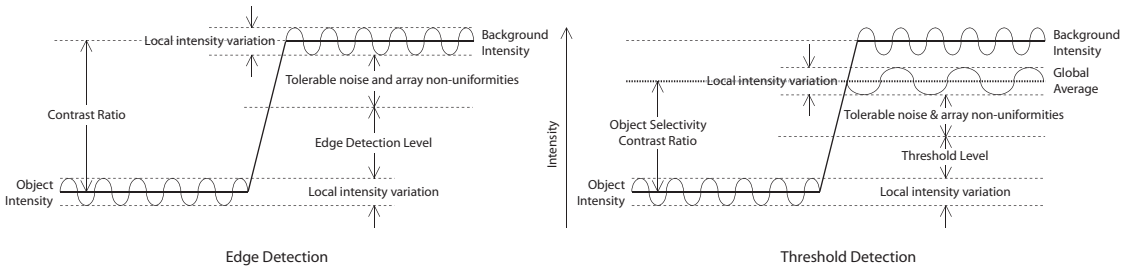


Figure 4.7: Acceptable noise margins for error-free binary edge detection and thresholding.

Analytical Robustness

This involves considering the intensity profiles and error tolerances of the input images and therefore determining the *noise margin* or *signal-to-noise ratio* for error-free binary feature extraction. Subsequently for a given image type, the optimum settings (threshold and edge levels) can be determined for maximum binary robustness and furthermore the suitability of the algorithm for different image types can be analysed. Finally, the relationship between distorted binary detection and erroneous centroid/sizing determination can be discussed.

Binary Feature Extraction Robustness

For edge detection (as defined in Eqns. 4.5 and 4.6) to be reliable in an input image consisting of (relatively) dark objects on a light background, the conditions specified in Eqns. 4.16 (for edge-detection) and 4.17 (for no edge-detection) need to be satisfied (see Fig. 4.7).

$$(I_{bg} - I_{obj}) - E_{offset} > I_{margin} + I_{noise} \tag{4.16}$$

$$E_{offset} > I_{margin} + I_{noise} \tag{4.17}$$

Where: $(I_{bg} - I_{obj})$ is the contrast difference of the objects to background level, E_{offset} is the minimum edge detection level, I_{margin} is the maximum object (and/or background) intensity variation and I_{noise} represents the maximum tolerable level of intensity variations (non-uniformities) in the array.

Assuming $I_{margin}(background) = I_{margin}(object)$ and I_{noise} is increased to the maximum allowable level such that Eqns. 4.16 and 4.17 become equalities, the optimum setting for E_{offset} can be specified for a given image type as expressed in 4.18.

$$E_{offset} = \frac{1}{2}(Ibg - Iobj) \tag{4.18}$$

However, the actual robustness to noise (Eqn. 4.17); causing erroneous edges to be detected, is in fact increased due to the post-edge-detection CONTOUR logic. The operation of this logic is to effectively screen out any noise triggering erroneous edges within object boundaries by only detecting edges outside object regions. Furthermore by performing the the thresholding operation on the narrow-field local averaged data, the object segmentation is also reliable against noise induced errors.

For threshold detection (as defined in Eqn. 4.4) to be reliable in an input image consisting of (relatively) dark objects on a light background, the conditions specified in Eqns. 4.19 (for thresholding) and 4.20 (for no thresholding) need to be satisfied (see Fig. 4.7).

$$(I_{av} - I_{obj}) - T_{offset} > I_{margin} + I_{noise} \tag{4.19}$$

$$(I_{bg} - I_{av}) + T_{offset} > I_{margin} + I_{noise} \tag{4.20}$$

Where: $(I_{av} - I_{obj})$ is the margin from average intensity to object intensity level, $(I_{bg} - I_{av})$ is the margin from background intensity to average intensity level, T_{offset} defines the threshold detection offset from average intensity, I_{margin} is the maximum object (and/or background) intensity variation and I_{noise} represents the maximum tolerable level of intensity variations (non-uniformities) in the array.

Assuming $I_{margin}(background) = I_{margin}(object)$ and I_{noise} is increased to the maximum allowable level such that Eqns. 4.19 and 4.20 become equalities, the optimum setting for T_{offset} can be specified for a given image type as expressed in 4.21. Depending on image content, T_{offset} may take a positive or negative value.

$$T_{offset} = \frac{1}{2}(2I_{av} - Iobj - Ibg) \tag{4.21}$$

To establish numerical data for the above described tolerances and intensity levels, analysing some sample images can provide typical values. By performing edge-detection and thresholding operations on sample red-blood-cell images (as those shown previously in Figs. 4.1 and 4.3), the resulting binary masks can be used to obtain statistical image content

Sample		RB00[1]	RB01	RB02	RB03	RB04	RB05
Global	Mean($\overline{x}$):	0.645	0.728	0.675	0.836	0.722	0.897
Background	Mean($\overline{x}$):	0.794	0.850	0.843	0.980	0.971	0.990
	Std(σ):	0.000[1]	0.023	0.029	0.030	0.044	0.015
Object	Mean($\overline{x}$):	0.409	0.624	0.497	0.698	0.440	0.741
	Std(σ):	0.000[1]	0.037	0.037	0.032	0.059	0.037
	Cover(%):	38.52	53.76	48.50	51.11	46.85	37.32
	Count(N):	56	380	291	52	46	41
Edge	E_{offset}:	0.193	0.113	0.089	0.069	0.141	0.078
	I_{noise}:	0.193	0.039	0.014	0.006	0.023	0.005
	$SNR(dB)$:	10.48[2]	25.53	33.66	42.88	29.94	45.08
Threshold	T_{offset}:	0.044	-0.009	0.005	-0.003	0.017	0.031
	I_{noise}:	0.197	0.039	0.104	0.074	0.164	0.082
	$SNR(dB)$:	10.30[2]	25.42	16.25	21.06	12.87	20.78

[1] Test image with ideal object and background uniformity.
[2] Indicates minimum possible SNR for a given contrast ratio.

Table 4.2: Example image analysis (red blood cells) for statistical spread in object and background intensity levels. This is used to determine the edge (E_{offset}) and threshold (T_{offset}) levels for optimum robustness.

data; extremely useful in determining the algorithmic robustness to that image type. Such an example analysis is provided in Table. 4.2.

This data is extracted by thresholding the various images at the global average level and clustering together all pixels within background and object regions. The statistical properties of each of these groups can therefore be computed and the specific feature variations can be compared for various images. What this data shows is that the robustness to the binary operations (edge detection and thresholding) is largely dominated by the variance of image object and background intensities in addition to array non-uniformities (considered as static noise). For input images with relatively high variance, the binary segmentation technique offers poor immunity to array non-uniformities (for example, only 5-10% noise margin for the red-blood cell images), whereas for images with relatively low variance, this technique can provide up to 30-40% noise margin.

Post-Binary Feature Extraction Robustness

The previous section has concentrated on optimising the feature extraction robustness based on the implemented binary operations and image content. However, a major objective in implementing distributed algorithms based on a biologically-inspired paradigm, is to capture the inherent robustness, defect-immunity and tolerance to ill-conditioned data of nature. Similarly, this algorithm is shown to substantially increase robustness beyond the expected analytical feature extraction limit.

This feature can be attributed to the parallel distributed processing forming multiple data flow paths coupled with data redundancy and compression. In the context of this algorithm, the data flow is initiated at the object contours and an inward fill is facilitated. This has the effect of compacting the amount of data being processed; for the "ring" of cells being processed at any time reduces with this inward propagation. Furthermore, this shrinking ring realises a many-to-one mapping and thus introduces massive data redundancy. For example, if an edge cell does not register a contour, or an internal cell does not register a threshold, the data propagation path simply tends to route itself around the erroneous data. This action is illustrated in Fig. 4.8. Similarly, this algorithm makes defective circuitry redundant provided it is not incorrectly reporting an initiation condition, eg. flagging a contour condition.

Due to the anticipated complexity in modelling this behaviour analytically, an alterna-

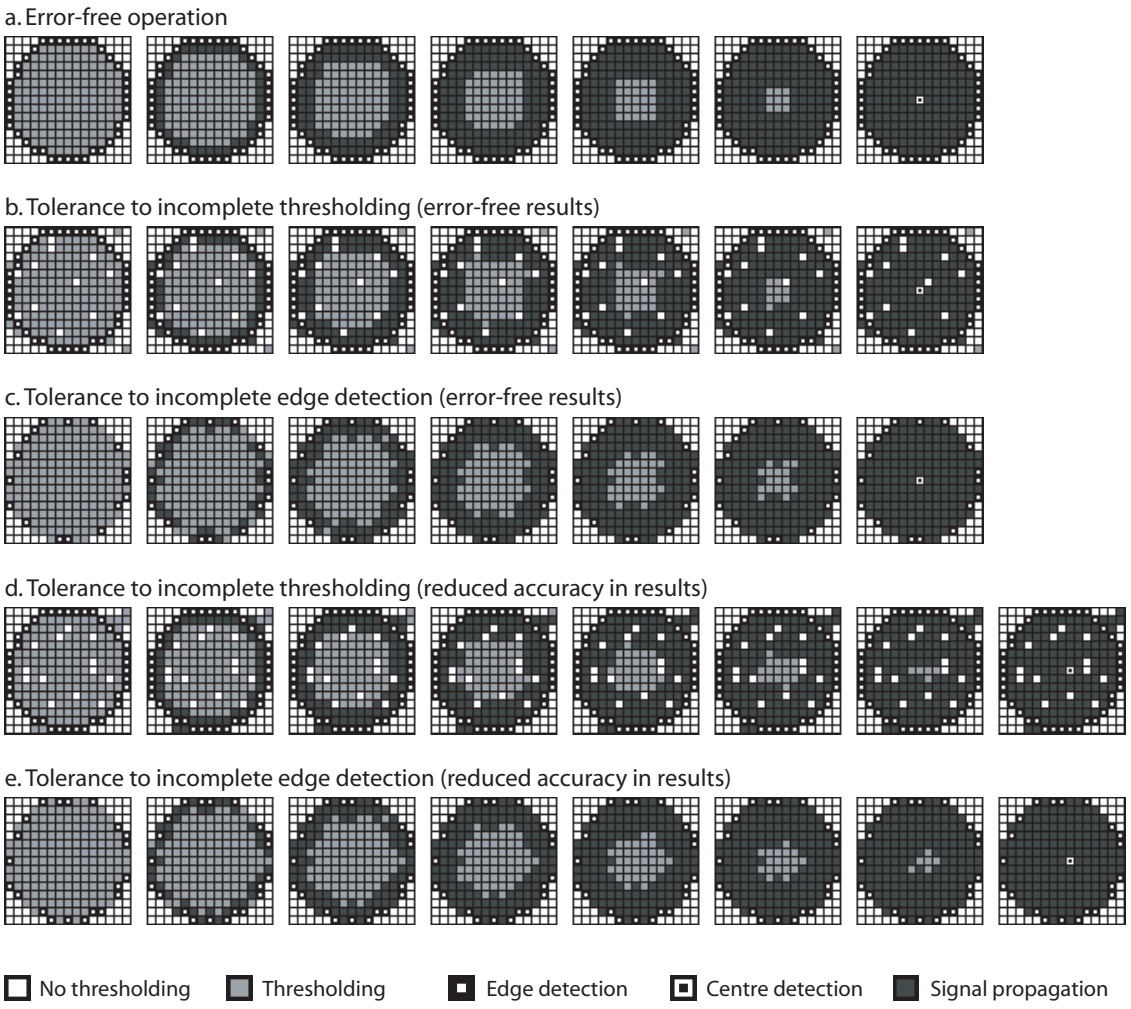


Figure 4.8: Simulated results demonstrating the algorithms inherent tolerance to erroneous (or incomplete) binary feature extraction. Shown are, response to: (a) perfect binary extraction, (b,d) to incomplete thresholding and (c,e) to incomplete edge detection.

tive technique to examine this added robustness is through statistical algorithmic simulations.

Simulated Statistical Robustness

This involves experimentally testing (simulating) the algorithm for various input images against the various expected sources of error (previously mentioned). To facilitate this, the input image data is pre-processed to include various levels of pixel-array non-uniformities and the algorithm is simulated to determine the outcome for each case. By normalising the results to the image content, this process can be repeated many times, collating statistical simulation results (in total 18,000 simulation runs are performed). The detailed statistical procedure that is taken is as follows:

- Input image: In total ten different input images have been used with different sizes and frequencies of circular objects. The images were chosen to have flat object and background texture in order to demonstrate the algorithmic response to array non-uniformities rather than to non-idealities in image feature uniformity.
- Contrast ratio: Each image is then altered to three set illumination levels; i.e. contrast ratios for object to background intensity ratios of 10:1, 4:1 and 3:2. Subsequently the average image intensity was adjusted to be at 50% maximum illumination level as to fairly capture effects to positive and negative non-uniformities. Image contrast ratio is the most crucial parameter when determining algorithmic robustness, therefore for fair comparison, only images of identical contrast ratio should be aggregated.
- Noise types: The sample images are subjected to three types of noise: Gaussian, speckle and salt pepper noise. These three types have been specifically chosen for their resemblance to various array non-uniformities in hardware fabrication. In particular (1) Additive (zero mean) Gaussian noise is used to simulate for photoreceptor offsets (fixed pattern noise), (2) Speckle noise represents gain and sensitivity non-uniformities of phototransduction and amplification elements and (3) Salt and Pepper noise is used to model dead or defective photoreceptors or in-pixel circuits with permanent low or high response.
- Noise power: Each type of noise is generated for twenty preset levels of noise power to cover 100% intensity spread, i.e. +/- 50% about the zero-noise intensity. It is these noise levels that shall be used as a reference to algorithmic robustness.

- Multiple simulation: In order to generate a sufficient amount of statistical data and achieve a conclusive trend, each simulation run is repeated using ten different noise sets.
- Algorithm settings: For each image/contrast ratio a single set of edge and object threshold levels are chosen based on the optimum dynamic range criteria expressed in Eqns. 4.18 and 4.21.
- Post simulation analysis: The generated results are averaged (for repeated runs), normalised to image content, i.e. object density (for different images) and then averaged again. The resulting data is then appropriately scaled to report the error in average object count and size for the three predefined contrast ratios.

These analysed results are illustrated in Fig. 4.9 for gaussian and speckle noise and Fig. 4.10 for salt and pepper noise. Concluding from these results, the good robustness of this algorithm is apparent. At no point does the algorithm cease to operate or "crash", however the accuracy becomes degraded with increased image noise. For typical fixed pattern noise levels [6] and medium contrast selectivity, an acceptable 2-5% inaccuracy can be observed. Whereas for images with high contrast ratios, higher accuracy can be expected. The algorithm also proves to be robust to defective pixel outputs, as indicated in the salt and pepper noise results.

4.2.8 Accuracy

Due to the binary nature of this centroiding/sizing algorithm, the centroid accuracy for regular objects, eg. circular objects, is limited to one pixel resolution. Similarly, radial precision can be also calculated to single pixel accuracy. However for irregular objects this centroid accuracy deteriorates. This algorithm is intended to provide a good estimate to centroid position and size, not a mathematically accurate centre-of-mass computation. For more precise object-centre determination, such a centre-of-mass/centre-of-gravity calculation can compute to multiple sub-pixel accuracy.

In general, the accuracy of perceptive vision processing tasks need not be highly accurate. For these are primarily used to divert attention of additional *parallel* processes to resolve the task to higher precision. For example, attention/saliency selection tasks typically have poor spatial resolution but good temporal resolution and are subsequently succeeded with processes of high spatial resolution.

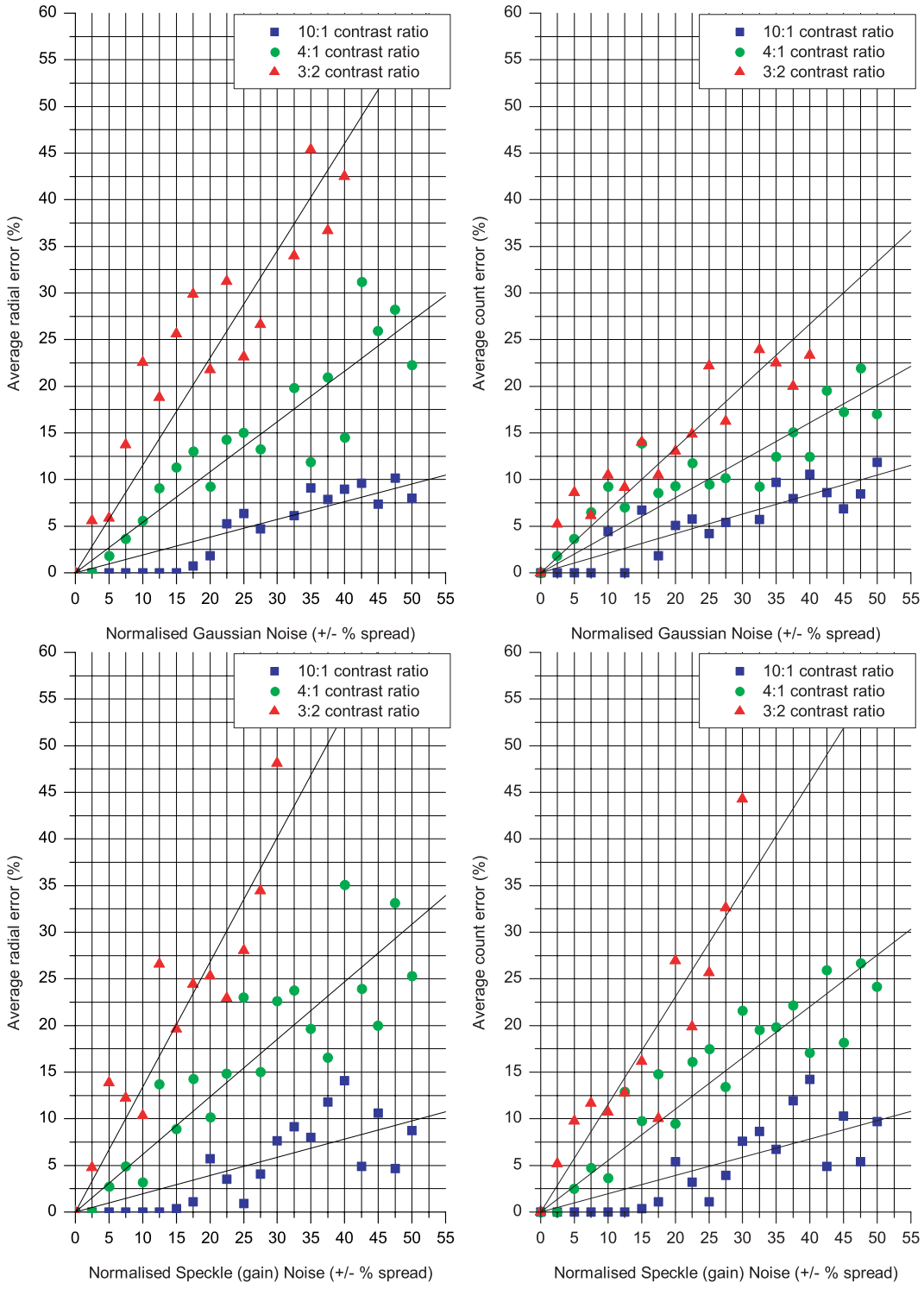


Figure 4.9: Statistical simulations to demonstrate robustness to array non-uniformities. Algorithmic response to additive spacial gaussian noise representing fixed pattern noise (top) and random speckle noise representing array gain/sensitivity non-uniformity (bottom).

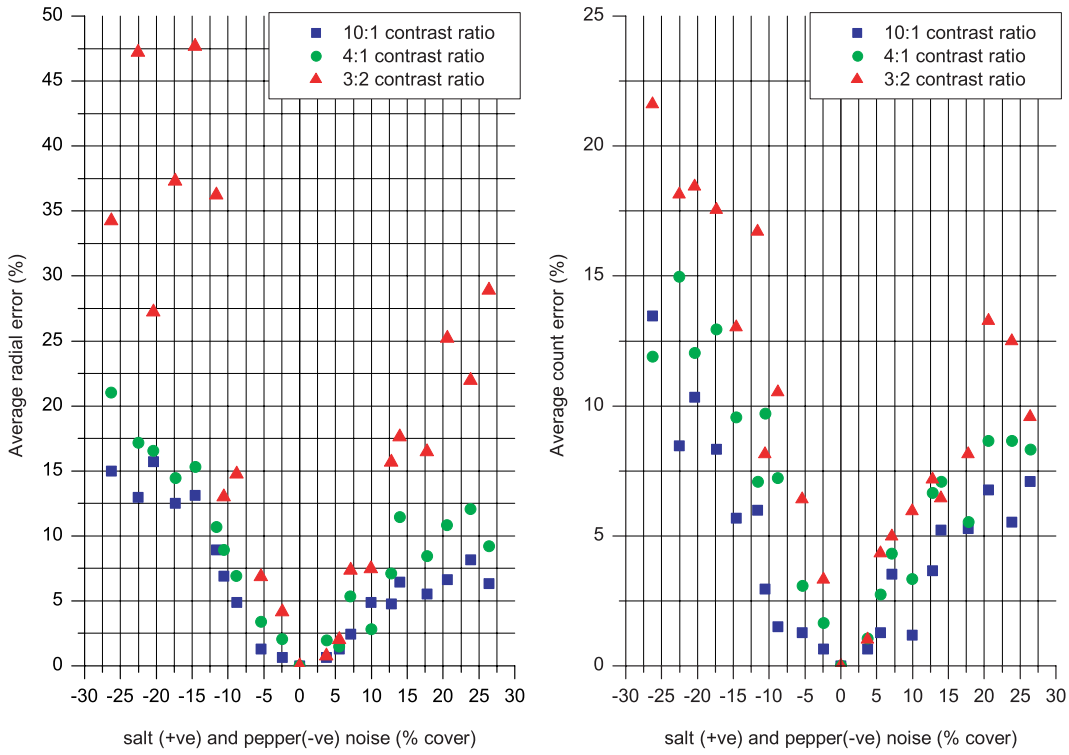


Figure 4.10: Statistical simulations to demonstrate robustness to array phototransduction/amplification defects. Algorithmic response to additive salt and pepper noise representing pixels with permanent low/high response.

4.3 A Bio-inspired Paradigm for Parallel Processing

By taking this specific (centroiding) distributed algorithm and analysing the connectivity and representation, a general architectural description can be derived. This method provides a formalisation to hardware implementation of intricate hybrid[3] structures based on common software techniques. Such techniques include recursion, erosion, dilation and backtracking; being only implementable in a sequential manner using conventional techniques. Distributed processing can offer significant performance advantages in computational efficiency, robustness, speed and processing capacity.

4.3.1 The Concept

The underlying concept behind this computational paradigm is to combine two different processing arrays. The front-end for signal acquisition and binary feature extraction and the back-end for binary spatiotemporal processing, the cross-connection being the extracted binary feature template. By dissociating the binary signal processing from the continuous signal conditioning in this way, advanced binary algorithms can be implemented within a true parallel architecture.

4.3.2 The Architecture

This distributed architecture can be implemented in several integrated sensor acquisition and processing applications requiring either one or two dimensional arrays.

This architecture can be therefore divided into the following five layers:

- Sensor acquisition layer (direct connectivity): This constitutes the sensor; input transducer including associated signal conditioning circuitry. For example, in an auditory system this could include a microphone and a pre-amplification stage. This layer could also include an off-array signal acquisition structure, for example interfacing to a microelectode array.
- Signal processing layer (direct and lateral connectivity): This includes continuous

[3]Hybrid refers to a system using multiple forms of data representation for parallel processing.

value spatiotemporal filtering including averaging, differentiating, integrating, normalising, etc. for preparation of binary extraction.

- Binary extraction layer (direct connectivity): Involves some thresholding functionality, i.e. a tuneable one-bit converter. This binary feature extraction produces digital signals (from analogue inputs) for driving the asynchronous logic array.
- Binary processing layer (direct and lateral connectivity): Forms a distributed asynchronous logic network, with parallel inputs for binary signal processing and subsequently high-level feature extraction. Advanced algorithms can include feedback to signal processing layer to achieve added adaptivity, tunability or accuracy.
- Array supervisory layer (lateral connectivity): Provides global inputs and control to all cells and handshakes processed information off-array. Communication with conventional processor can providing start data or provide search functionality.

This generalised distributed processing architecture is illustrated in Fig. 4.11.

4.3.3 Neurobiological analogy

The general architecture presented in fact shares several similarities to neurobiology and in particular the visual system. The following analogies can be made for each of the above mentioned layers:

- Sensor acquisition layer → Retinal photoreceptor layer: These neurons generate electrical signals on absorption of photons of light and through local amplification produces inputs suitable for the subsequent signal processing.
- Signal processing layer → Retinal bipolar, horizontal and amacrine layers: These biological cells form receptive fields producing parallel spatiotemporally-processed visual streams for subsequent image-feature extraction.
- Binary extraction layer → Retinal ganglion cell layer: Through "integrate-and-fire" functionality this neural layer aggregates signal amplitude, converting to the temporal domain as discrete pulse-frequency data, subsequently transmitted to the brain.
- Binary processing layer → Cortical primary visual cortex: V1 cells are believed to have orientation selective functionality using localised interactions facilitating signal propagation [7] in a similar manner to that intended in this binary processing layer.

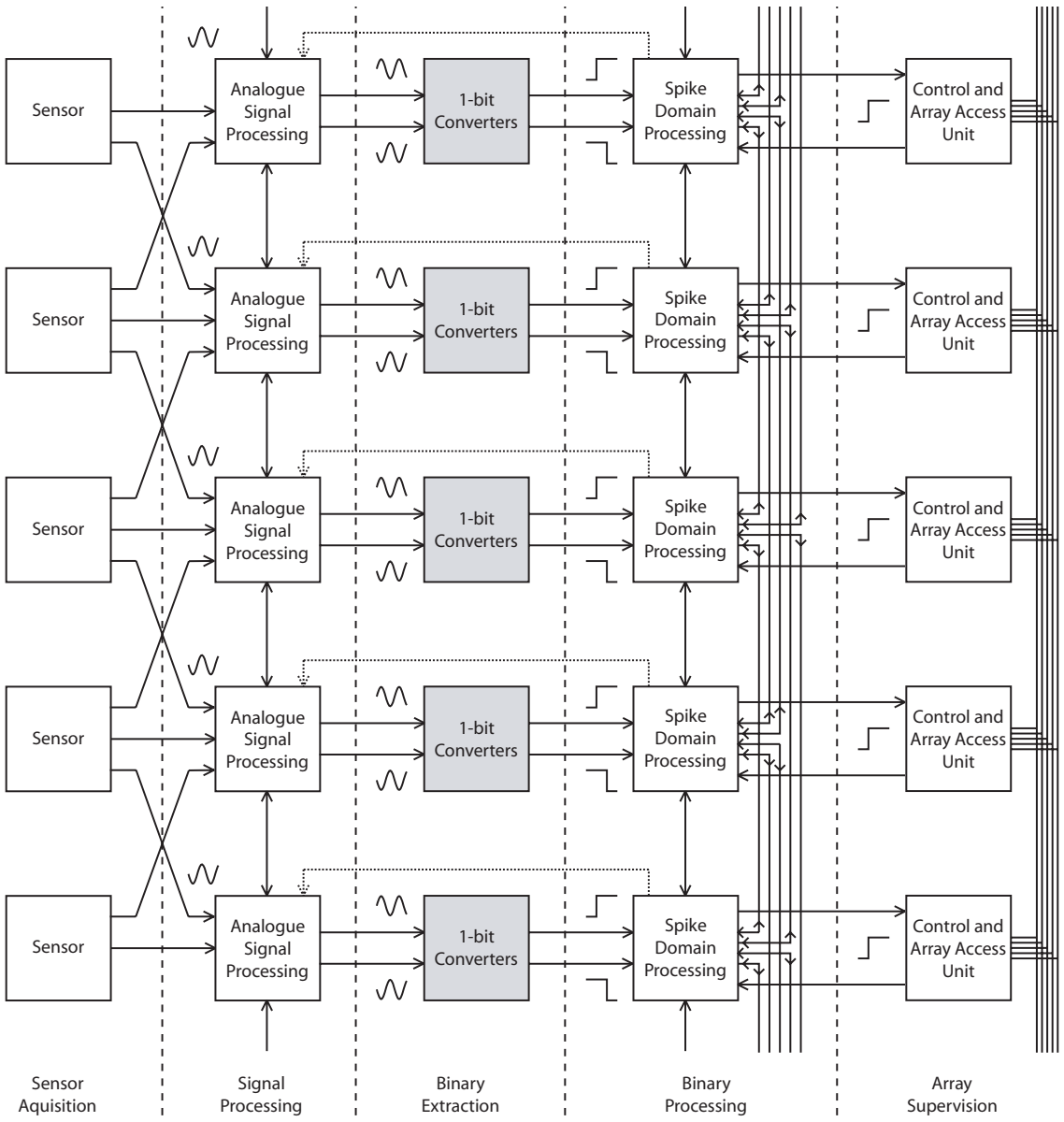


Figure 4.11: The generalised distributed processing architecture; a one dimensional array illustrating the various functional layers and interconnectivity. Increased performance and/or added functionality (eg. localised gain control, noise-shaping, oversampling, etc) could be realised through closed loop conversion mechanisms (dotted arrows), although these would be purely synthetic as oppose to biologically-inspired.

- Array supervisory layer → Higher order cortical processes: Although structurally different, these provide high-order perceptive functionality such as saliency and attention selection, very much performing a supervisory or control role over lower-order processes.

4.3.4 Implementation

This architecture is best suited for implementation in CMOS technology. Possessing phototransductive (photodiodes) or mechanical acoustic (MEMS) elements, CMOS technology can offer sensor integration; critical for such distributed processing platforms. Furthermore, implementing this architecture using MOSFET devices operated in the weak inversion region, coupled with asynchronous digital logic provides the ideal combination for power efficient realisation. Power efficiency is extremely important when tessellating several thousand (or more) cells on a single silicon substrate.

Application

A reconfigurable *array processor* would perhaps be the most suitable system based on this architecture. Since logic is easily made reconfigurable and front-end binary feature extraction functions can be made general, an FPGA-like vision processor could be useful for a variety of applications. On the other hand, a reconfigurable system would be far from being universally applicable. For well defined tasks with specific algorithm a custom implementation would achieve the optimum performance. The difficulty therefore lies in developing the distributed algorithm to be implemented, as these are generally less-intuitive to formulate.

4.4 Emerging technologies

A number of leading research establishments and semiconductor foundries are currently investing in and developing next generation 3D CMOS technologies [8] [9]. These expect to provide multiply stacked substrates with interconnecting vias; in a similar manner to metal layers in current processes. Such a technology would have a huge impact on the imaging and vision chip community. For example the exposed substrate layer can be specifically devoted and tailored to imaging; with low substrate doping and near 100% fill factor. Subsequently,

the "underground" layers could contain the processing electronics in true 3D retinomorphic arrangement [10].

Although this emerging technology is believed to revolutionise the semiconductor industry as a whole, it will only be truly beneficial to a very small proportion. For example, most high-performance digital processing systems currently limited due to power dissipation constraints will not be assisted by advances in such technologies. Therefore, foreseeing ahead it would be advisable to develop those circuits and systems that will benefit from such future technologies. It is expected, systems involving distributed architectures; particularly involving vision applications, will be amongst those to thrive the most in these emerging technologies.

4.5 Summary

This chapter introduces the concept of the bio-pulsating contour reduction algorithm. This is a parallel, distributed algorithm performing asynchronous object recognition breaking the bottleneck of traditional, sequential von Neumann based computational paradigm. The globally asynchronous scheme is regulated by employing data-generated local synchronisation, increasing computational efficiency and improving the signal-to-noise ratio. By incorporating the processing in the front end, the bandwidth requirements have been reduced by at least four orders of magnitude.

Through developing a software equivalent technique, the computational complexity has been estimated and by comparison to state-of-the-art DSP technology a target computational benchmark has been determined. Furthermore, it has been established that conventional techniques are preferable for off-line processing applications, providing superior accuracy. On the other hand, distributed techniques are more suited to realtime, high-speed and high-resolution processing.

Analysis and simulation into algorithmic robustness have identified two crucial factors: image content and array non-uniformities. Analysis has determined robustness to be highly dependant on image type and content. Subsequently a method for defining the suitability of the algorithm to a certain image type has been outlined. Furthermore, statistical simulations have tested the algorithm to different types and levels of spatial noise and concluded very good robustness to array non-uniformities.

Finally a distributed processing platform has been outlined; based on the bio-pulsating contour reduction algorithm. This extends the core architecture by generalising the various sub-blocks, realising a platform suited to hardware implementation of a wide-range of distributed algorithms.

References

[1] T. G. Constandinou, T. S. Lande and C. Toumazou, "Bio-pulsating architecture for object-based processing in next generation vision systems," *IEE Electronics Letters*, vol. 30, no. 16, pp. 1169–1170, 2003.

[2] H. DeMan, F. Catthoor, G. Goosens, J. Vanhoof, J. Van Meerbergen, S. Note and J. Huisken, "Architecture-driven synthesis techniques for VLSI implementation of DSP algorithms," *Proceedings of the IEEE*, vol. 78, no. 2, pp. 319–335, 1990.

[3] J. M. Rabaey and M. Pedram, *Low Power Design Methodologies*. Kluwer Academic Publishers, 1995.

[4] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele and A. Vandecappelle, *Custom Memory Management Methodology - Exploration of Memory Organisation for Embedded Multimedia System Design*. Kluwer Academic Publishers, 1998.

[5] Texas Instruments, "TMS320C5000 Power Efficient Digital Signal Processors (DSP's) Overview," *http://dspvillage.ti.com/docs/allproducttree.jhtml?pageId=C5*, 2005.

[6] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts and J. Bogaerts, "A Logarithmic Response CMOS Image Sensor with On-Chip Calibration," *IEEE Journal of Solid-state Circuits*, vol. 35, no. 8, pp. 1146–1152, 2000.

[7] Z. Li, "Visual segmentation by contextual influences via intracortical interactions in primary visual cortex," *Network: Computation in Neural Systems*, vol. 10, no. 2, pp. 187–212, 1999.

[8] J. Baliga, "Chips go vertical [3D IC interconnection]," *IEEE Spectrum*, vol. 41, no. 3, pp. 43–47, 2004.

[9] R. Islam, C. Brubaker, P. Lindner and C. Schaefer, "Wafer level packaging and 3D interconnect for IC technology," *IEEE/SEMI Conference and Workshop on Advanced Semiconductor Manufacturing*, pp. 212–217, 2002.

[10] J. Georgiou and A. G. Andreou, "A mixed analog/digital asynchronous processor for network models of cortical computation," *9th International Conference on Cognitive and Neural Systems*, 2005.

Chapter 5

Photodiodes in Modern Deep Sub-Micron CMOS Technology

5.1 Introduction

A major objective in implementing any vision processing system is monolithic integration of imaging elements with processing circuits. Traditionally, the limiting factor has been large area overhead for circuit implementation as past CMOS technologies (i.e. of larger feature sizes) have generally yielded either low-resolution systems or systems of limited in-pixel processing complexity. In modern technologies (i.e. sub-350nm minimum feature size) however, this trend has been reversed and although the in-pixel circuits scale in size and include higher complexity, the photodetection elements tend not to scale [1, 2] so favourably. It is therefore a challenge in developing any vision processing system in modern deep submicron CMOS technology is to achieve photodetection elements of performance comparable with those available in past technologies.

This chapter reviews silicon-based phototransduction in CMOS technology specifically concerning p-n photodiode implementation, developing an analytical model intended for deep submicron technologies. Then a range of fabricated devices in a given technology ($0.18\mu m$ CMOS) are presented and the their measured results discussed and analysed, particularly concerning deep submicron technology related issues. Several key factors are then identified and a generic set of design rules outlined for photodiode optimisation. Finally, different interface topologies are reviewed leading to a novel bio-inspired spiking photoreceptor with a scheme for adaptable/tradable dynamic range, spatial and temporal resolution.

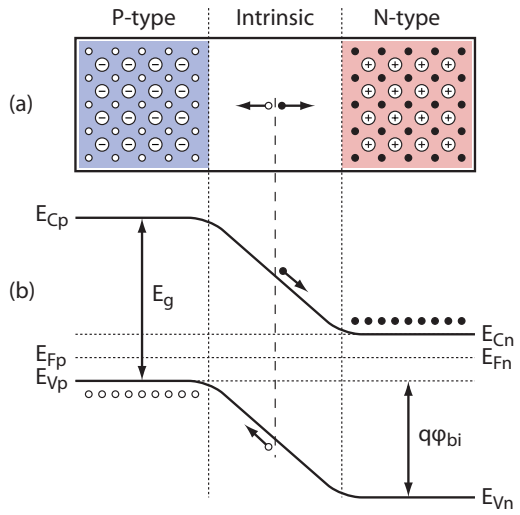


Figure 5.1: The basic P-I-N diode under zero external bias illustrating: (a) cross-section and (b) energy band diagram.

5.2 Photodiode Modelling

Much work has already gone into modelling and analysing silicon-based photodiode behaviour [3, 4, 5, 6, 7, 8, 9, 10]. This section aims to develop a comprehensive yet concise model; useful in pre-fabrication design and optimisation of CMOS based P-N junction photodiodes, particularly targeted towards implementation in deep submicron technologies.

5.2.1 Silicon-based Phototransduction

Traditionally, silicon-based photodiodes have been implemented using p-type/intrinsic/n-type (P-I-N) structures. Phototransduction occurs when incident photon energy is absorbed within the intrinsic region causing an electron-hole pair to split and collected resulting in a photo-detectable current. This process can be described by its energy band diagram, shown in Fig. 5.1.

Incident light radiation; of photon energy (hv) being greater than E_g ($E_g(Si) = 1.1eV$) will result in excitation of an electron from the valence band (E_v) to the conduction band (E_c). This process manifests itself as an electron-hole pair splitting and subsequently as it will consume exactly 1.1eV, any excess energy will be dissipated thermally.

As the absorption requires the excitation of valence band electrons of a certain energy

and these are finite per unit volume, it is therefore dependant on the thickness of the semiconductor. Given an incident photon flux of Φ, penetrating a thickness of δx; causes a change in photon flux of $\Delta\Phi$, the *absorption coefficient* is defined as:

$$\alpha = -\frac{\Delta\Phi}{\Phi\Delta x} \tag{5.1}$$

Where α is the absorption coefficient, $\Phi = (\lambda/hc)(P_{opt}/A)$ is the photon flux, P_{opt} is the incident optical power, A is the cross-section area and the negative sign represents an attenuation. Integrating Eqn. 5.1 for illumination of constant wavelength, forms the *Beer-Lambert Law* describing the transmitted photon flux to decrease exponentially with depth:

$$\Phi(x) = \Phi_0 e^{-\alpha x} \tag{5.2}$$

Where Φ_0 is the initial photon flux (i.e. at x=0) and x is the depth. Alternatively, expressing the absorption coefficient as a function of wavelength yields the relationship:

$$\alpha \approx \frac{2\omega n_i}{c} = \frac{4\pi n_i}{\lambda} \tag{5.3}$$

Where λ is the wavelength of incident radiation and n_i is the imaginary part of the materials refractive index. This expression in fact reveals a significant relationship; that the longer the wavelength, the deeper it can penetrate into a given material. Furthermore, the *penetration depth*; derived from Eqn. 5.2 is defined as the depth a given wavelength can penetrate until it is attenuated by 63% $(1/e)$ its original value. This is expressed as: $y = \alpha^{-1}$.

5.2.2 The P-N Junction Photodiode

As standard deep-submicron CMOS technologies typically do not include process steps for creating defined intrinsic layers for fabricating P-I-N photodiode structures, parasitic P-N junctions are generally used for phototransduction. At the P-N junction, majority carrier diffusion gives cause to a depletion layer being formed having pseudo-intrinsic properties and therefore being suitable for photo-absorption; generating electron-hole pairs. This diffusion is due to the majority carrier concentration gradient with holes diffusing from the p-type

region to the n-type and electrons vice versa. As a result, the region near the P-N junction is depleted of majority carriers and hence the term depletion region (Fig. 5.2a,b).

Depletion Layer [3]

As the space charge region (depletion layer) has no free charge carriers, the depleted n-type region has a net positive charge and the depleted p-type region has a net negative charge (Fig. 5.2c). This gives rise to an internal junction "built-in" voltage, expressed in Eqn. 5.4.

$$\phi_{bi} = \frac{kT}{q} ln \frac{N_A N_D}{n_i^2} \tag{5.4}$$

Where ϕ_{bi} is the junction built-in voltage, $kT/q(= \phi_t)$ is the thermal voltage, N_A and N_D are the impurity acceptor and donor atom concentrations and n_i is the intrinsic semiconductor concentration.

Once determining this built-in potential (Fig. 5.2e), the width of the depletion region can be determined.

$$x_w = \sqrt{\frac{2\varepsilon_0 \varepsilon_{r(Si)}}{q} \left(\frac{1}{N_A} + \frac{1}{N_D} \right) (\phi_{bi} - v_b)} \tag{5.5}$$

Where x_w is the depletion width, ε_0 is the permittivity of vacuum, $\varepsilon_{r(Si)}$ is the relative permittivity of silicon and v_b is the applied bias. Consequently, the depletion region can be made to increase in width by applying a reverse bias; the higher the bias, the wider the depletion region.

Furthermore, this depletion region gives rise to an internal electric field (Fig. 5.2d) due to the static charge differential. This is crucial in separating photo-generated electron-hole pairs and thus collecting the photocurrent. The magnitude of this internal electric field at the abrupt junction interface is expressed in Eqn. 5.6.

$$E_0 = -\frac{qN_D x_{wn}}{\epsilon_0 \epsilon_r} = -\frac{qN_A x_{wp}}{\epsilon_0 \epsilon_r} \tag{5.6}$$

Where E_0 is the maximum internal electric field and x_{wn} and x_{wp} are the depletion region widths in the n- and p-type regions respectively.

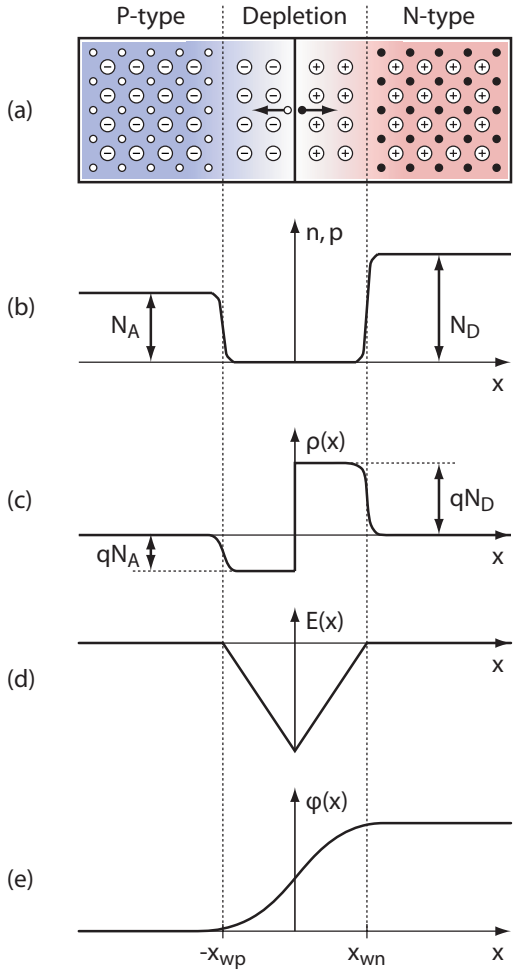


Figure 5.2: The basic P-N junction diode under zero external bias illustrating: (a) cross-section (b) n/p concentration profile (c) space charge density (d) electric field and (e) internal potential

Photocurrent Density [11]

The electron-hole pair generation rate can be directly derived from the Beer-Lambert law (Eqn. 5.2) giving:

$$G(x) = -\frac{d\Phi(x)}{dx} = \alpha\Phi_0 e^{-\alpha x} \tag{5.7}$$

The photocurrent density (J_{photo}) consists of two components: the drift current (J_{drift}); due to carriers generated within the depletion region and diffusion current ($J_{diffusion}$); due to carriers generated outside the depletion region. Assuming an initial photon flux (Φ_0) at the edge of the depletion region, the expression for the drift current component becomes:

$$J_{drift} = -q\int_0^{x_w} G(x)dx = q\left(\Phi_{x_w} - \Phi_0\right) = -q\Phi_0\left(1 - e^{-\alpha x_d}\right) \tag{5.8}$$

Where x_d is the depletion layer width in the axis of the incident light radiation. For a vertical junction this would be equal to the depletion layer width (i.e. $x_d = x_w$), however for a lateral junction this would be equal to the junction depth (i.e. $x_d = x_z$).

The diffusion current consists of two sub-components; the reverse diffusion current density under dark conditions (J_{dark}) and the photogenerated contribution from the substrate ($J_{diff,vert}$ and $J_{diff,horiz}$ for vertical and lateral diffusion). The expression for dark current density can be derived from:

$$J_{dark} = J_s = \frac{qD_n n_{po}}{L_n} + \frac{qD_p p_{no}}{L_p} \tag{5.9}$$

Where J_s is the saturation current density, $L_n = \sqrt{D_n\tau_n}$ and $L_p = \sqrt{D_p\tau_p}$ are the electron and hole diffusion lengths, $D_n = \phi_t\mu_n$ and $D_p = \phi_t\mu_p$ are the n- and p-type diffusion constants, μ_n and μ_p are the electron and hole mobilities and n_{p0} and p_{n0} are the equilibrium minority carrier densities in the n- and p-type regions.

Assuming that either the n- or p-type region is much more heavily doped, i.e. in the case of a diode to the p-substrate (n+/p-), $N_D \gg N_A$ and that $V_r > 3\phi_t$, where V_r is the reverse bias, Eqn. 5.9 can be reduced to:

$$J_{dark} \simeq q\left(\sqrt{\frac{D_n}{\tau_n}}\frac{n_i^2}{N_A} + \frac{n_i x_w}{\tau_g}\right) \tag{5.10}$$

Where τ_g is the generation lifetime. [3]

The photogenerated diffusion contribution from the substrate can be expressed as one-dimensional diffusion equations for vertical and horizontal junctions, given in Eqns.5.11 and 5.12 respectively.

$$J_{diff,vert} = -q\Phi_0 \left(\frac{\alpha L_p}{1 + \alpha L_p} \right) e^{-\alpha x_w} \tag{5.11}$$

$$J_{diff,horiz} = -q\Phi_0 \left(1 - \frac{x_{xpi} - x_x}{2L_p} \right) \tag{5.12}$$

Where L_p is the minority carrier diffusion length in the substrate, x_{xpi} is the total photodiode width (x-pitch), x_x is the junction width (lateral) and x_j is the junction depth. The geometric design parameters are illustrated in Fig. 5.4.

The total photocurrent can therefore be expressed separately for the vertical (Eqn. 5.13) and horizontal (Eqn. 5.14) junctions, each consisting of its drift and diffusion components:

$$I_{photo,vert} = x_x y_x \left(J_{drift,vert} + J_{diff,vert} + J_{dark} \right) \tag{5.13}$$

$$I_{photo,horiz} = 2x_w \left(x_x + x_y \right) J_{drift,horiz} + \left(x_{xpi} x_{ypi} - x_x x_y \right) J_{diff,horiz} + 2x_j \left(x_x + xy \right) J_{dark} \tag{5.14}$$

Where x_{ypi} is the total photodiode length (y-pitch).

These expressions do not account for the fact that charge carriers generated near the semiconductor surface are likely to recombine due to surface effects. As a result, these expressions are likely to evaluate higher current densities for short wavelength radiation than real devices are expected to measure.

5.2.3 Photodiode Efficiency

Having defined the photocurrent density for a P-N junction region, the photodiode efficiency can be determined by including the device geometric and physical design parameters.

The two main factors affecting the overall photodiode (or quantum) efficiency are the optical and dimensional efficiencies. The optical interface will generally attenuate and redistribute the incident light radiation, whereas the internal silicon P-N junction structure can be designed to optimally absorb and detect the remaining photon flux.

Optical Transmission [12]

A crucial factor in determining a photodiodes overall efficiency is the transmission through its optical interface. In CMOS technologies this generally includes a metal mask on top metal layer; shielding non-photodetecting elements from incident light radiation and a transparent passivation/field-oxide layer acting as the air/coating/substrate interface. A typical profile of a modern CMOS technology is illustrated in Fig. 5.3.

Considering a substrate with refractive index n_s on top of which a dielectric layer (of refractive index n_c and thickness t) is coated, light of wavelength λ strikes the coating surface from the air ($n_0 = 1$). The air/coating(SiO_2) and coating(SiO_2)/substrate(Si) interface reflectances are given in Eqns. 5.15 and 5.16.

$$r_{0C} = \frac{n_C - n_0}{n_C + n_0} \tag{5.15}$$

$$r_{CS} = \frac{n_S - n_C}{n_S + n_C} \tag{5.16}$$

Where r_{0C} is the complex amplitude reflection coefficient for the air/coating interface and r_{CS} is the complex amplitude reflection coefficient for the coating/substrate interface.

The propagation of the incident light within the coating introduces a phase shift, described by:

$$\phi_C = 4\pi \frac{t}{\lambda_0} n_C \tag{5.17}$$

Where ϕ_C is the optical phase shift, λ_0 is the incident radiation wavelength in vacuum and t is the coating thickness.

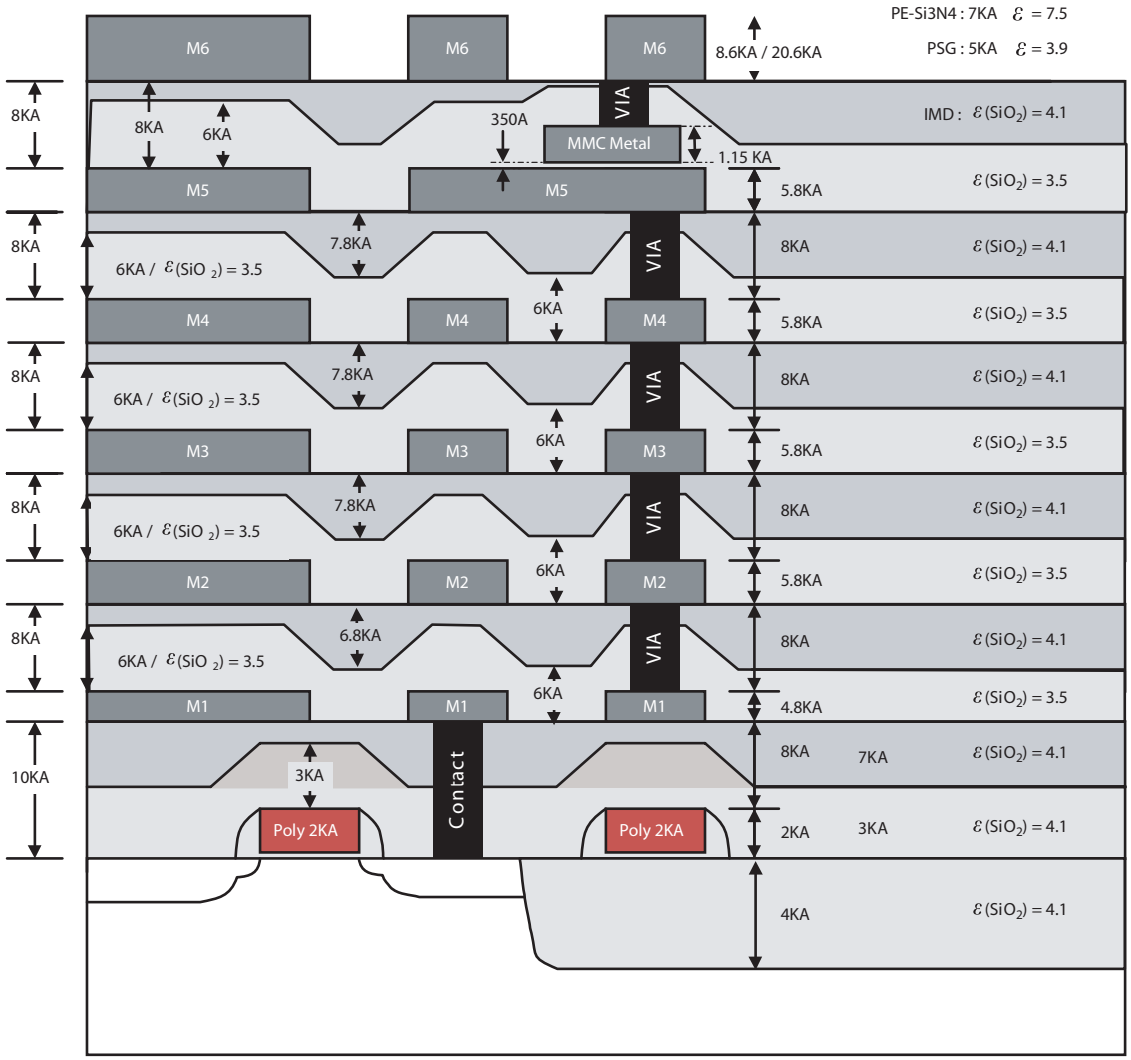


Figure 5.3: Cross section of a typical modern deep submicron CMOS technology showing the stacked metal layers with insulating dielectrics and optical transmission path (right).

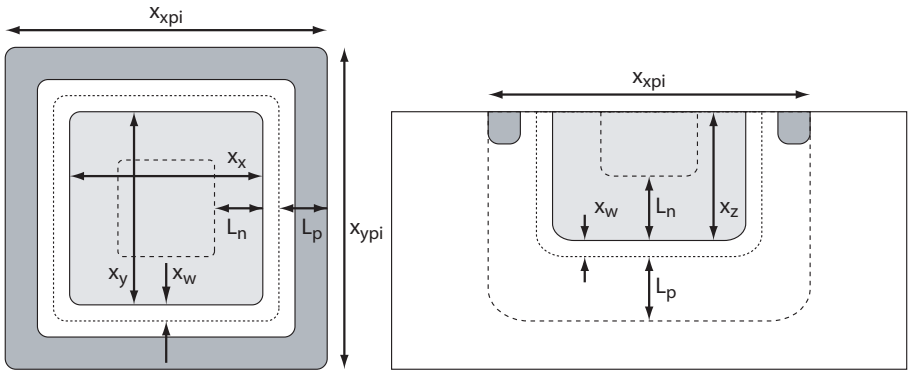


Figure 5.4: Basic geometric dimensions (design and technology/bias defined) for a single junction device representing the surface (left) and cross-section (right) views.

Assuming that all the materials are lossless and that the substrate is semi-infinite, the reflection coefficient can be expressed as [13, 8]:

$$r = \left| \frac{r_{0C} + r_{CS} e^{i\phi_C}}{1 + r_{0C} r_{CS} e^{i\phi_C}} \right|^2 = 1 - \left(\frac{\left(1 - r_{0C}^2\right)\left(1 - r_{CS}^2\right)}{1 + r_{0C}^2 r_{CS}^2 + 2 r_{0C} r_{CS} cos\phi_C} \right) \tag{5.18}$$

Where r is the effective reflectance of air/coating/substrate optical interface. Subsequently the optical efficiency (η_0) is defined as the transmission, i.e. $\eta_o = T = 1 - r$.

Geometric Substrate Utilisation

The second factor affecting overall photodiode efficiency, for a given illumination area is the geometric substrate utilisation; dependant on depletion layer and diffusion region volume, depth and exposed surface.

Any three dimensional p-n photodiode structure can be modelled as two parallel devices; one representing the vertical junctions and one representing the horizontal (lateral) junctions. The basic geometric dimensions for a single junction device is illustrated in Fig. 5.4

Vertical pn-junction efficiency can be determined from the derived photocurrent expression (Eqn. 5.13) by including the absorption effect for a junction beneath the surface. Similarly, horizontal pn-junction efficiency can be determined based on the derived lateral photocurrent expression (Eqn. 5.14) assuming the remaining substrate region (i.e. outside

the junction area but still within the photodiode "allocation") can contribute to the lateral diffusion current.

External Quantum Efficiency

The external quantum efficiency defines the amount of electron-hole pairs contributing towards the photocurrent for every incident photon. The external quantum efficiencies have been determined separately for each surface of the pn-junction, combined to give the resultant photocurrent.

The external quantum efficiency for a vertical pn-junction (Eqn. 5.19) is the classic expression often used in photodiode designs with relatively low edge effect (high vertical to lateral ratio).

$$\eta_{vert} = (1 - R)\,\zeta \left(1 - \frac{e^{\alpha x_w}}{1 + \alpha L}\right) e^{-\alpha x_z} \tag{5.19}$$

Where η_{vert} is the external and ζ is the internal quantum efficiency for the vertical surface of a pn-junction, $1 - R$ is the optical efficiency and x_z is the junction depth.

The lateral external quantum efficiency is defined using separate expressions for the lateral drift (Eqn. 5.20) [11] and diffusion (Eqn. 5.21) contributions.

$$\eta_{horiz,drift} = (1 - R)\,\zeta \left(1 - \frac{e^{\alpha(x_z + x_w)}}{1 + \alpha L}\right) \tag{5.20}$$

$$\eta_{horiz,diff} = (1 - R)\,\zeta \left(1 - \frac{x_{xpi} - x_x}{2L}\right) \tag{5.21}$$

Where $\eta_{horiz,drift}$ and $\eta_{horiz,diff}$ are the external (drift and diffusion) and ζ is the internal quantum efficiency for the horizontal (lateral) surface of a pn-junction.

These quantum efficiency expressions can then be combined by considering the proportion of incident photon flux on each junction depletion (or diffusion) region, yielding Eqn. 5.22.

$$\eta_{eff} = \left(\frac{(x_x x_y)\,\eta_{vert}}{x_{xpi} x_{ypi}}\right) + \left(\frac{2x_w\,(x_x + xy)\,\eta_{horiz,drift}}{x_{xpi} x_{ypi}}\right) + \left(\frac{(x_{xpi} x_{ypi} - x_x x_y)\,\eta_{horiz,diff}}{x_{xpi} x_{ypi}}\right) \tag{5.22}$$

Where x_x is the junction width, x_y is the junction length, and x_w is the depletion width. This assumes that all exposed substrate area is utilised either by vertical or horizontal pn junction depletion layer.

Photoresponse

Subsequently, the device responsivity provides an expression for the photoresponse per unit irradiance by including the electronic charge and photon energy.

$$R = \frac{\eta_{eff} \cdot q}{hv} = \frac{\eta_{eff} \cdot \lambda}{1.24} \tag{5.23}$$

Where R is the responsivity and λ is the incident radiation wavelength (in μm).

$$I_\phi = R \cdot Q_\lambda + J_{dark}\left[x_x x_y + 2x_z\,(x_x + x_y)\right] \tag{5.24}$$

Where I_ϕ is the photocurrent generated by incident (monochromatic) radiation of Q_λ irradiance (W/m^2) and J_{dark} is the dark current density defined in Eqn. 5.10.

The photodiode capacitance can be reduced to the junction depletion layer capacitance including both base and sidewall junctions; expressed in Eqn. 5.25.

$$C_j = \frac{\varepsilon\left[x_x x_y + 2x_z\,(x_x + x_y)\right]}{x_w} \tag{5.25}$$

Where C_j is the reverse bias junction capacitance. As previously mentioned, the depletion width can be modulated by reverse bias (Eqn. 5.5), thus the higher the reverse bias, the wider the depletion region and consequently the smaller the junction capacitance.

5.3 Photodiode Characterisation

5.3.1 Technology

The test technology for the photodiode characterisation is UMC 0.18μm 1P6M MM/RF CMOS [14].

This process has the following standard features; useful utilisation of which can assist the photodiode designer to engineer optimally performing devices:

- Substrate processing: non-epitaxial p-substrate with masks for n-well, p-well, t-well (within n-well) and n++/p++ diffusion formation. Doping profile available on request (crucial for determining depletion region).
- Passivation: High density plasma (HDP), poly-silicon glass (PSG), passivation enhanced silicon nitride (PESIN).
- Interconnect: Pitch 0.42μm for poly, 0.48μm for metal 1, 0.58μm for metal 2 to 5, 2.2μm for metal 6. Thickness 2.0kA for poly, 4.8kA for metal 1, 5.8kA for metal 2 to 5, 20.6kA for metal 6.

5.3.2 Device Design

A total of fourteen (14) different test (photodiode) devices have been designed and fabricated in the above mentioned technology (mentioned in section 5.3.1). These can be grouped into three categories with the following objectives:

- Single junction topologies (4 devices): To test single junction vertical pn-junction structures and thus validate vertical junction model. Furthermore the effect of substrate doping and reverse bias can be examined without influence of side effect (due to side-walls). These devices have the side-walls (optically) shielded to ensure measured photoresponse is only a measure dependant on vertical junction efficiency. Dark current however will be influenced by the side-wall areas contributing diffusion current.
- Multiple paralleled (lateral) junction topologies (7 devices): To test multiple vertical and lateral pn-junction structures and thus empirically determine relative vertical and horizontal junction efficiencies. Furthermore, by testing different well/substrate

collection schemes, the influence of diffusion current and recombination effects can be observed.

- Multiple stacked (vertical) junction topologies (3 devices): To test spectral response due to selective absorption based on photon energy, i.e. wavelength, at different junction depths. This will provide verification if such structures are feasible in standard CMOS deep-submicron technologies. It is important to note that these devices are not intended for use as phototransistors and will be tested such that all junctions remain reverse biased.

All these fabricated device topologies are illustrated (diagrammatically) in Figs. 5.5 (single junction devices) and 5.6 (multiple junction devices). Furthermore, actual chip microphotographs of these structures are provided in Fig. 5.7. The design parameters for these presented photodiode structures are provided in Table 5.1.

5.3.3 Device Measurements

Details of the equipment setup and measurement procedure for electrical and optoelectronic characterisation are provided in Appendix E.

As all devices are built in the same process, they all have a similar optical efficiency and therefore any differences are due to different semiconductor structures, not due to optical transmission transfer.

Electrical Response

Measured electrical characteristics yielding a series of current-voltage relationships (per device) under different irradiance levels are illustrated in Figs. 5.8 and 5.9. The presented measurements are for a maximum irradiance of $0.45mW/cm^2$ at $\lambda = 550nm$. Furthermore, dark current characteristics are included to quantify dynamic range; to the tested optical signal power and can subsequently determine responsivity (discussed next).

Furthermore, a summary of extracted electrical characteristics is provided in Table 5.2. Measured dark current values are higher than expected partly due to equipment limitations, i.e. the minimum reliably measurable current is of the order of 100fA (for the equipment used). Other parameters extracted are open-circuit voltage, short-circuit current, fill factor

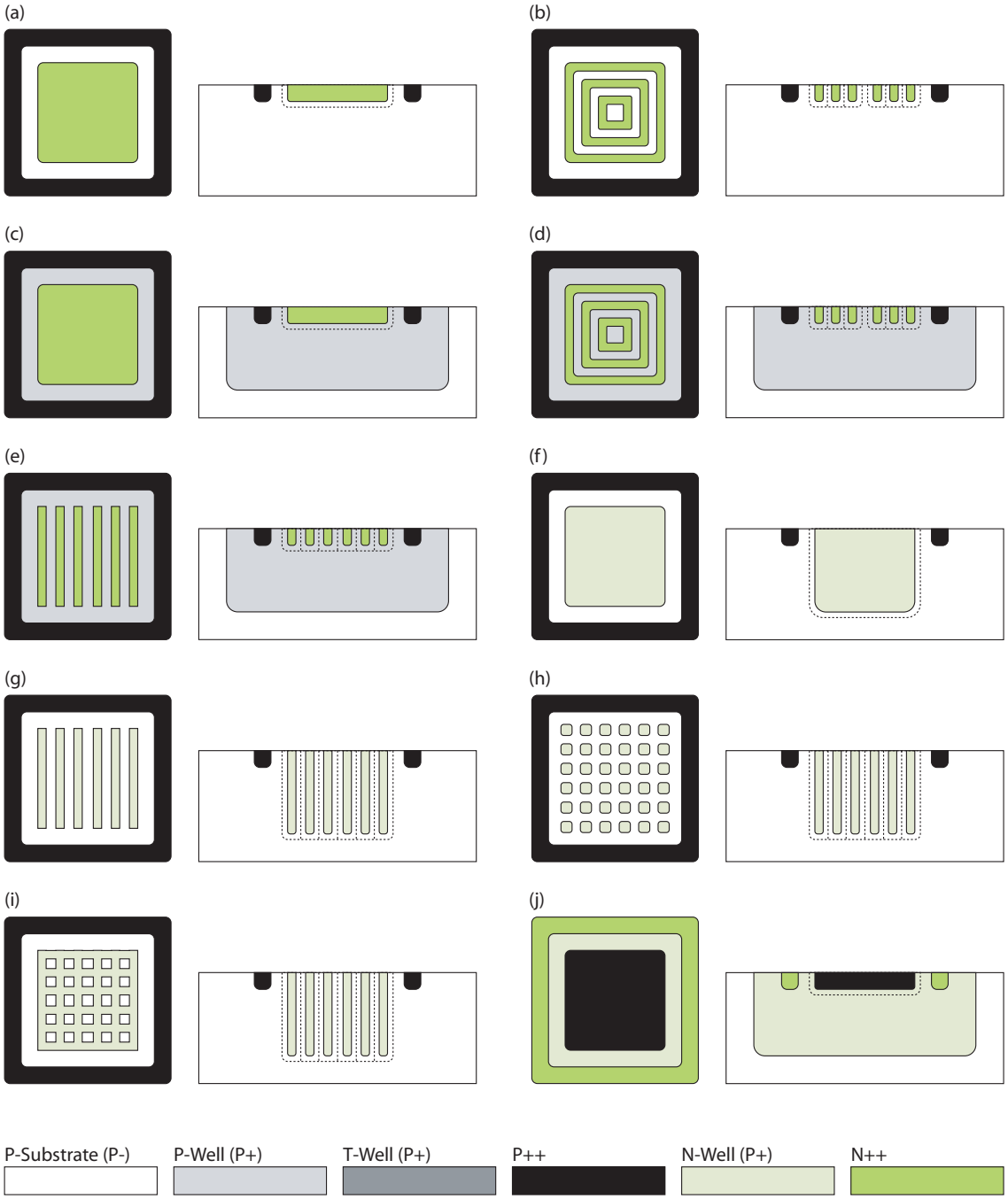


Figure 5.5: Various single-junction photodiode structures fabricated and tested in 0.18μm CMOS. Illustrated are the surface and cross-section views of the following structures: (a) n++/p-substrate (b) n++ rings/p-substrate (c) n++/p-well (d) n++ rings/p-well (e) n++ strips/p-well (f) n-well/p-substrate (g) n-well strips/p-substrate (h) n-well grid/p-substrate (i) n-well mesh/p-substrate and (j) p++/n-well. All devices are sized 30μm × 30μm and illustrations not to scale.

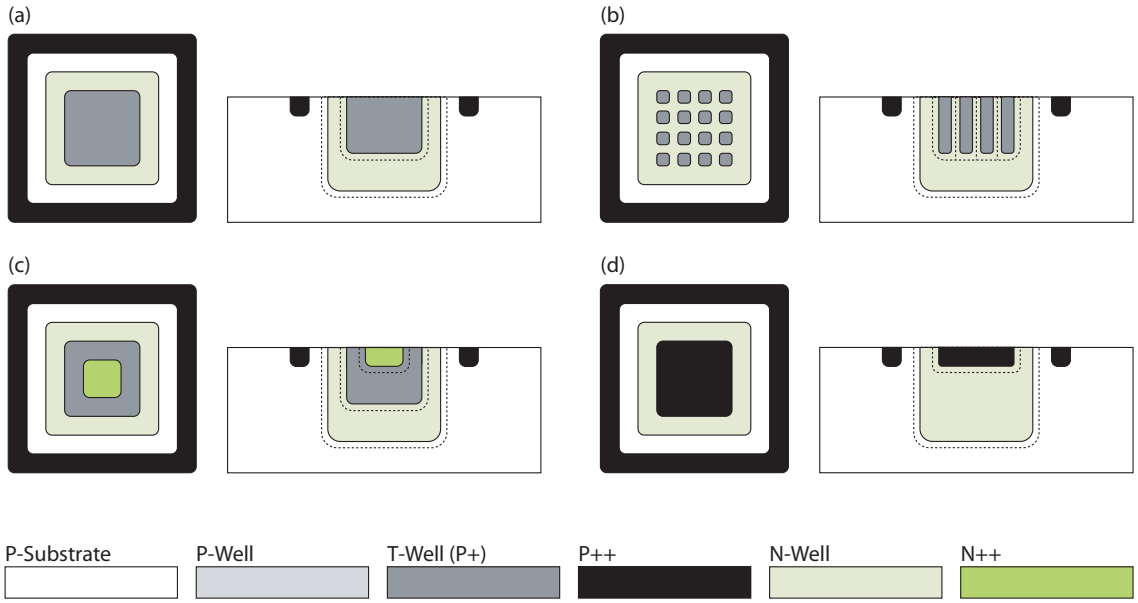


Figure 5.6: Various multi-junction photodiode structures fabricated and tested in $0.18\mu m$ CMOS. Illustrated are the surface and cross-section views of the following structures: (a) t-well/n-well/p-substrate (b) t-well grid/n-well/p-substrate (c) n++/t-well/n-well/p-substrate and (d) p++/n-well/p-substrate. All devices are sized $30\mu m \times 30\mu m$ and illustrations not to scale.

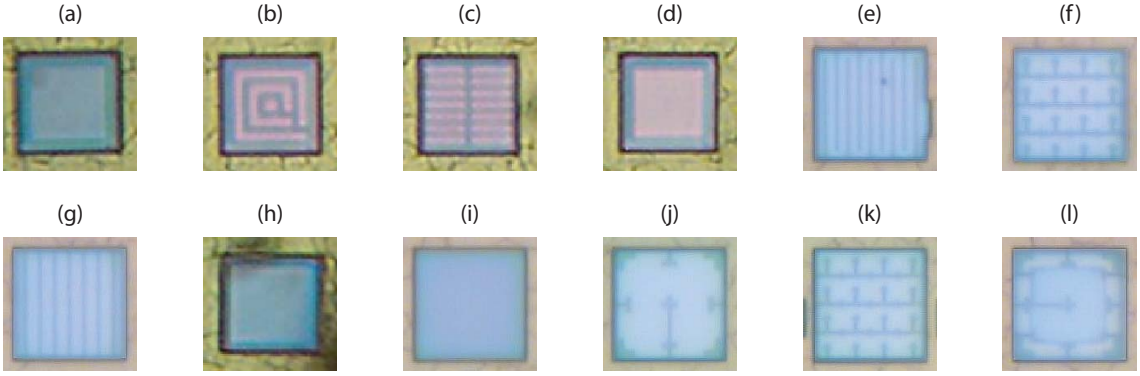


Figure 5.7: Microphotographs of photodiode structures fabricated and tested in $0.18\mu m$ CMOS. Illustrated are: (a) n++/p-well (b) n++ rings/p-well (c) n++ strips/p-well (d) n-well/p-substrate (e) n-well strips/p-substrate (f) n-well grid/p-substrate (g) n-well mesh/p-substrate, (h) p++/n-well, (i) p++/n-well grid, (j) t-well/n-well/p-substrate (k) t-well grid/n-well/p-substrate and (l) n++/t-well/n-well/p-substrate. All devices are sized $30\mu m \times 30\mu m$.

Junction structure	Active area [1]	Active perimeter[1]	Junction doping[2]	Junction depth
Single junction devices (single isolated terminal, shared substrate)				
n++/p-sub	$673.4\mu m^2$	-	$10^{19}/8\times10^{14}$	$0.18\mu m$
n++ rings/p-sub	$71.3\mu m^2$	$300.0\mu m$	$10^{19}/8\times10^{14}$	$0.18\mu m$
n++/p-well	$673.4\mu m^2$	-	$10^{19}/10^{17}$	$0.18\mu m$
n++ rings/p-well	$71.3\mu m^2$	-	$10^{19}/10^{17}$	$0.18\mu m$
n++ strips/p-well	$91.4\mu m^2$	$381.5\mu m$	$10^{19}/10^{17}$	$0.18\mu m$
n-well/p-sub	$673.4\mu m^2$	-	$2\times10^{17}/8\times10^{14}$	$1.80\mu m$
n-well strips/p-sub	$322.8\mu m^2$	$274.7\mu m$	$2\times10^{17}/8\times10^{14}$	$1.80\mu m$
n-well grid/p-sub	$411.4\mu m^2$	$384.0\mu m$	$2\times10^{17}/8\times10^{14}$	$1.80\mu m$
n-well mesh/p-sub	$324.7\mu m^2$	$921.6\mu m$	$2\times10^{17}/8\times10^{14}$	$1.80\mu m$
p++/n-well	$509.8\mu m^2$	$92.0\mu m$	$10^{19}/2\times10^{17}$	$0.20\mu m$
p++ strips/n-well	$381.0\mu m^2$	$789.1\mu m$	$10^{19}/2\times10^{17}$	$0.20\mu m$
Double junction devices (two isolated terminals, shared substrate)				
(t-well/n-well/p-sub) device				
t-well/n-well	$446.8\mu m^2$	$83.7\mu m$	$6\times10^{17}/2\times10^{17}$	$1.20\mu m$
n-well/p-sub	$578.3\mu m^2$	$95.9\mu m$	$2\times10^{17}/8\times10^{14}$	$1.80\mu m$
(t-well grid/n-well/p-sub) device				
t-well grid/n-well	$431.3\mu m^2$	$384.0\mu m$	$6\times10^{17}/2\times10^{17}$	$1.20\mu m$
Triple junction devices (three isolated terminals, shared substrate)				
(n++/t-well/n-well/p-sub) device				
n++/t-well	$329.5\mu m^2$	$71.9\mu m$	$10^{19}/6\times10^{17}$	$0.18\mu m$
t-well/n-well	$432.0\mu m^2$	$81.0\mu m$	$6\times10^{17}/2\times10^{17}$	$1.20\mu m$
n-well/p-sub	$567.3\mu m^2$	$91.9\mu m$	$2\times10^{17}/8\times10^{14}$	$1.80\mu m$

[1] Including only optically exposed active junction area/perimeter.
[2] Assuming abrupt junction; valid due to high junction doping differential.

Table 5.1: Design parameters (process defined and geometric) for the various test (photodiode) structures.

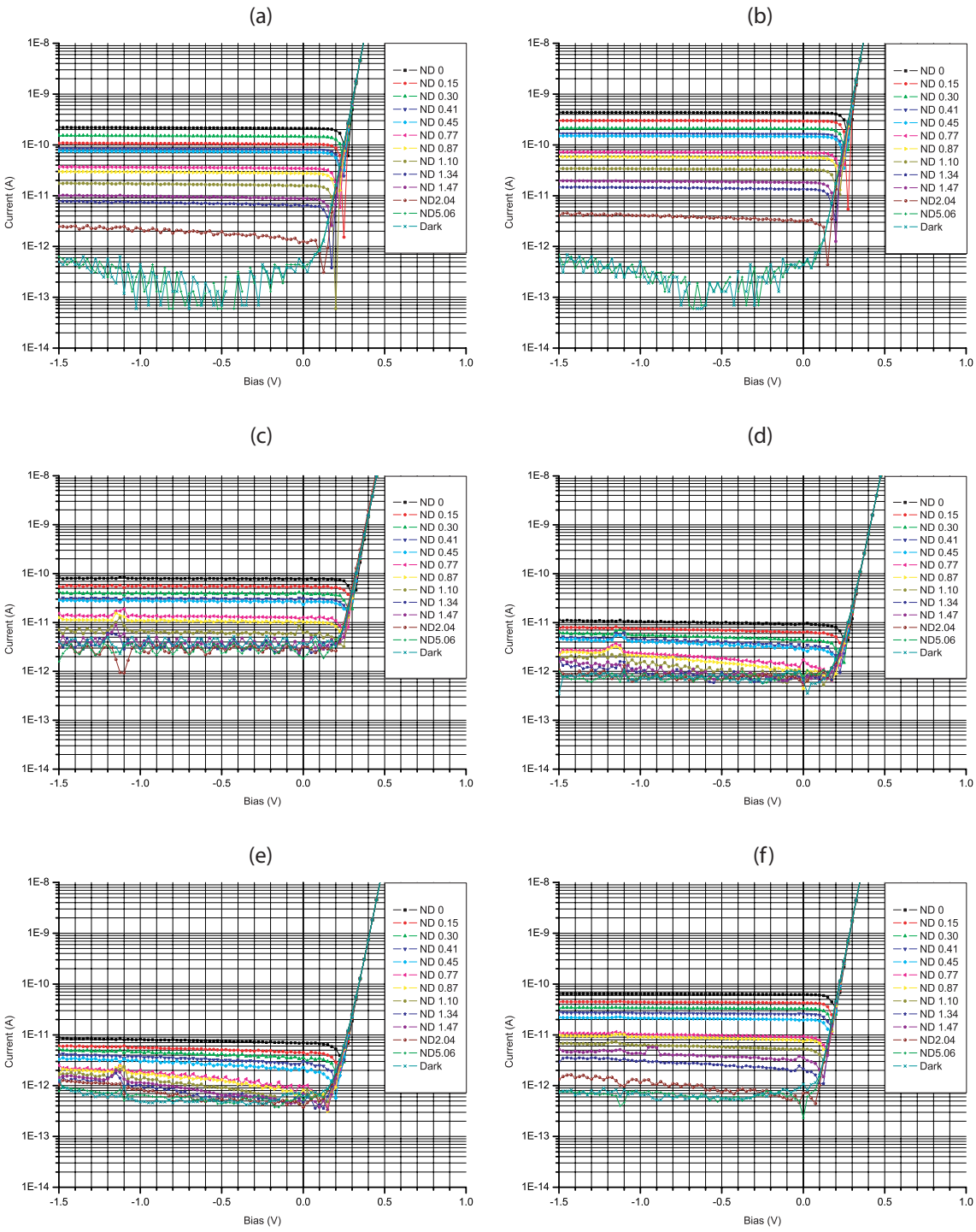


Figure 5.8: Measured IV characteristics of various test (photodiode) structures (using calibrated light source: λ=550nm, P_{max}=0.45mW/cm^2). Shown are the characteristics for: (a) n++/p-substrate (b) n++ rings/p-substrate (c) n++/p-well (d) n++ rings/p-well (e) n++ strips/p-well and (f) n-well/p-substrate.

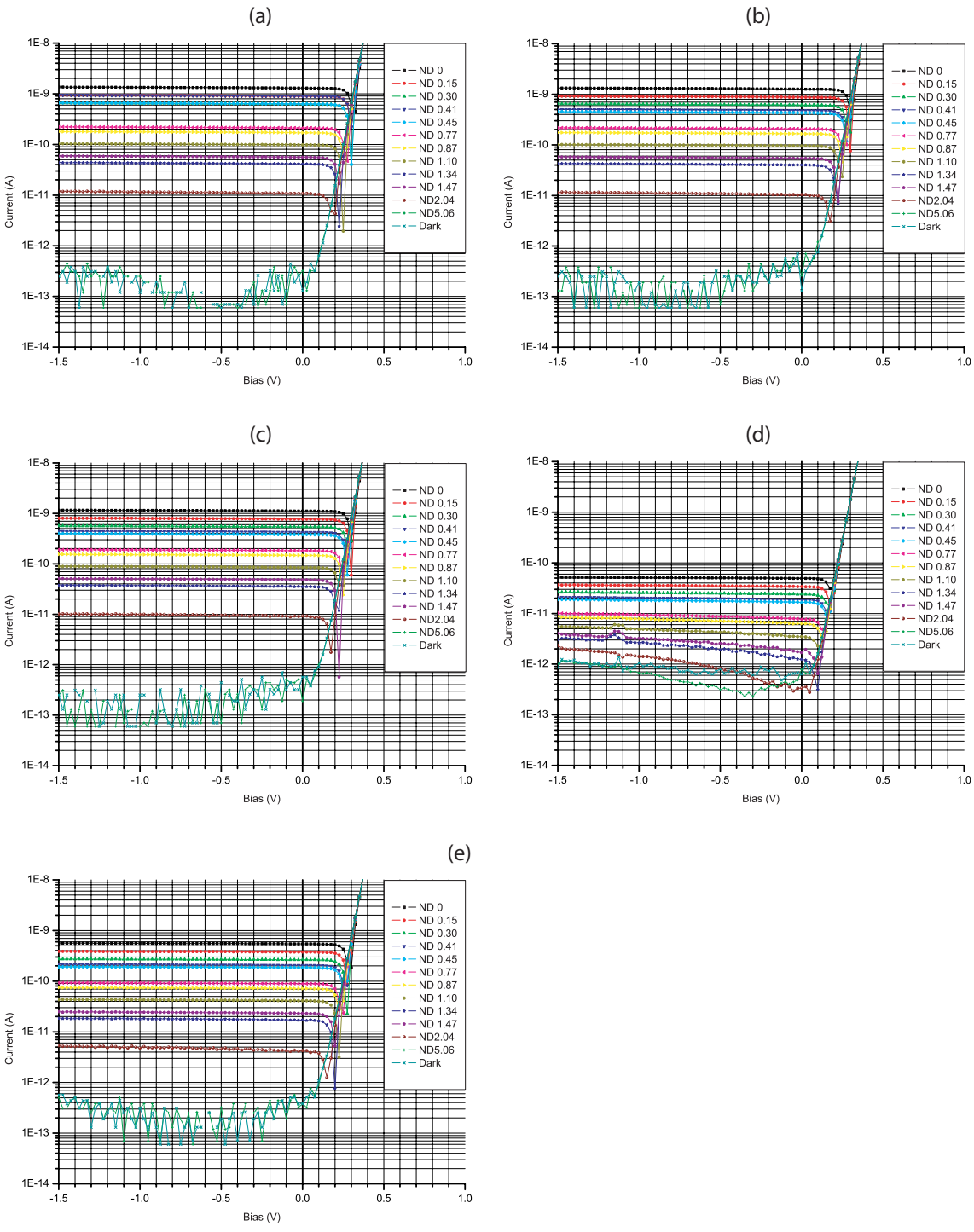


Figure 5.9: Measured IV characteristics of various test (photodiode) structures (using calibrated light source: λ=550nm, P_{max}=0.45mW/cm^2). Shown are the characteristics for: (a) n-well strips/p-substrate (b) n-well grid/p-substrate (c) n-well mesh/p-substrate (d) p++/n-well and (e) p++ strips/n-well.

Junction structure	Open-circuit voltage	Short-circuit current	Fill factor	Shunt resistance[1]	Dark current[2][3]	Dynamic Range [3][4]
n++/p-sub	0.284V	211pA	64.52%	$75G\Omega$	440fA	53.6dB
n++ rings/p-sub	0.317V	428pA	65.05%	$50G\Omega$	440fA	59.8dB
n++/p-well	0.148V	29.98pA	25.81%	$4.4G\Omega$	28.9fA	60.3dB
n++ rings/p-well	0.385V	180.3pA	69.35%	$13G\Omega$	49.9fA	71.2dB
n++ strips/p-well	0.356V	42pA	55.17%	$40G\Omega$	580fA	37.2dB
n-well/p-sub	0.295V	270pA	60.65%	$12G\Omega$	680fA	52.0dB
n-well strips/p-sub	0.336V	1.30nA	67.84%	$21G\Omega$	130fA	80.0dB
n-well grid/p-sub	0.341V	1.27nA	65.60%	$12G\Omega$	130fA	67.0dB
n-well mesh/p-sub	0.345V	1.11nA	63.85%	$20G\Omega$	570fA	65.8dB
p++/n-well	0.277V	120pA	57.66%	$60G\Omega$	840fA	43.1dB
p++ strips/n-well	0.311V	544pA	66.16%	$33G\Omega$	380fA	63.1dB

[1] Measured over $0 \leq V_{bias} \leq 50mV$
[2] Dark current measured at: $V_{bias} = 0V$
[3] Limited by resolution of current measurement: 10^{-13}A
[4] Dynamic range for given light power density: 0.45mW/cm^2

Table 5.2: Measured electrical characteristics for the various test (photodiode) structures. Light source is calibrated at: P_{light}=0.45mW/cm^2, λ=550nm.

and shunt resistance. The open-circuit voltage V_oc is identified as the voltage across the illuminated device at zero current. This is useful when the device is to be used in photovoltaic mode. Similarly, the short-circuit current I_{sc}, is the current through the illuminated device if the voltage across it is zero. This gives a measure of the device responsivity and subsequently its quantum efficiency. The fill factor (not to be confused with *surface area fill factor*) of the device is defined as the ratio of the maximum power of the device to the product of the open-circuit voltage and short-circuit current.

Light Intensity Response

For a maximum irradiance of $0.45mW/cm^2$ at $\lambda = 550nm$ (under zero bias), the measured responsivity (photocurrent versus irradiance) is illustrated in Fig. 5.10. This has been extracted from the IV curves presented previously.

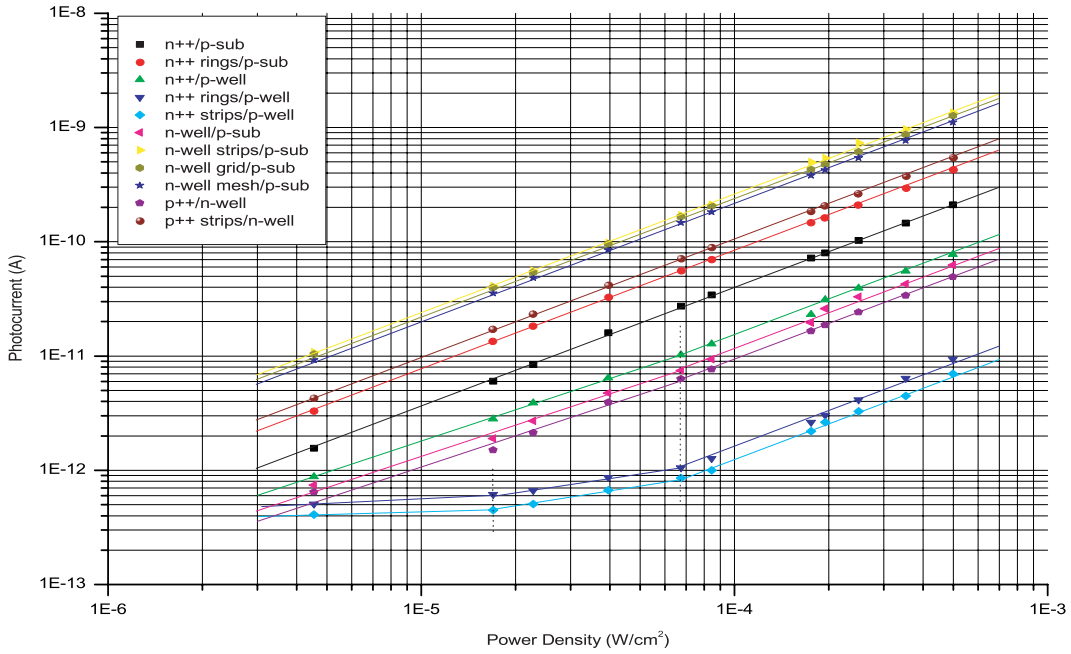


Figure 5.10: Measured responsivities of various test (single junction photodiode) structures (using calibrated light source: λ=550nm, P_{max}=0.45mW/cm^2).

The results show excellent linearity over 2-3 orders of magnitude, although the photodiodes complete linear range is expected to be 4-5 orders of magnitude. The reason the responsivity has not been measured to cover a wider range is due to a limited set of Neutral Density (ND) filters within the light source attenuator (See Appendix E). Specifically, there was no intermediate filter between ND2.04 and ND5.07, where the latter results in photocurrent levels in the region of 10-500fA; at the low-end current measurement resolution of the test equipment setup.

Spectral Response

Spectral characterisation is conducted to a calibrated irradiance for incident light of wavelength $\lambda = 350nm$ to $\lambda = 750nm$ at $\Delta\lambda = 5nm$ increments.

- Spectral efficiency (single junction devices): Measured results for spectral photoresponse, responsivity and absolute external quantum efficiency are presented in Figs. 5.11, 5.12 and 5.13 respectively. Generally the devices utilising minimally doped semiconductor junctions tend to most efficient, i.e. those devices collecting onto the substrate.

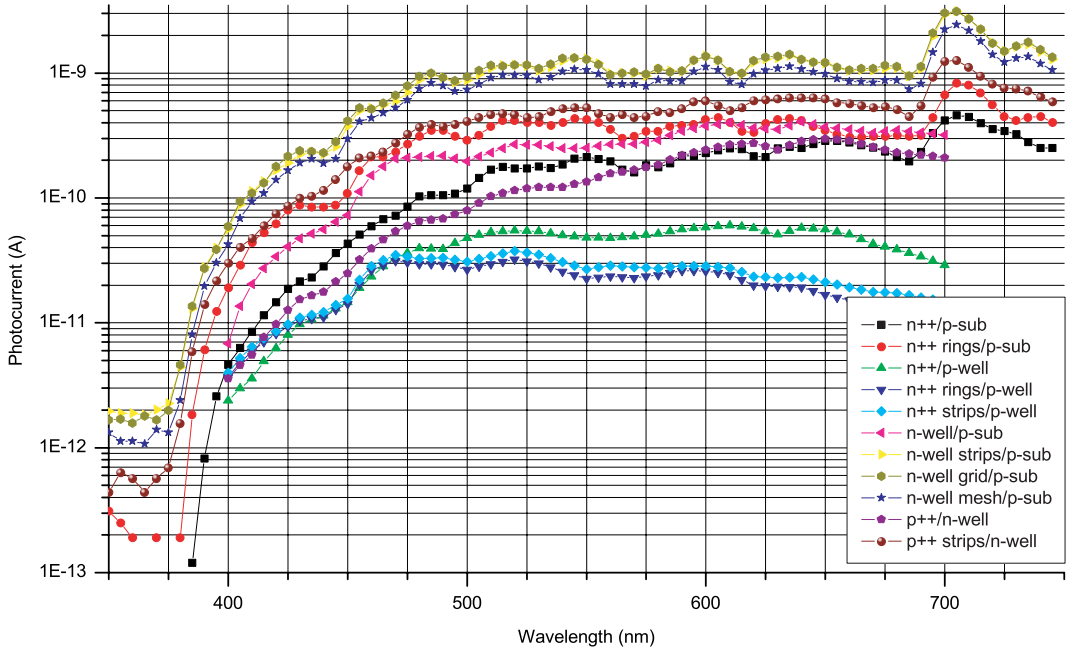


Figure 5.11: Measured spectral photoresponse of single junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$).

Furthermore, devices with increased lateral junction area show to significantly improve device efficiency. Devices with single (vertical) junctions are the least efficient, however they have other favourable properties (discussed below).

- Spectral efficiency (multiple junction devices): Measured results for spectral photoresponse, responsivity and absolute external quantum efficiency are presented in Figs. 5.14, 5.15 and 5.16 respectively. Generally the deeper the junction, the higher the responsivity as also observed in the single junction devices. Consequently long wavelength response tends to exhibit higher efficiencies than shorter wavelengths.

- Spectrally selective devices (single junction): The normalised quantum efficiency results for devices that show spectral selectivity are presented in Fig. 5.17. Generally the devices fabricated with single vertical junctions are the most spectrally selective. The only exceptions are multi-junction (lateral) devices with relatively shallow junctions to a relatively highly doped bulk.

- Spectrally selective devices (multiple junction): The normalised quantum efficiency results for multi-junction devices that show spectral selectivity are presented in Fig. 5.18. Both devices tested demonstrate good spectral selectivity. Generally as expected,

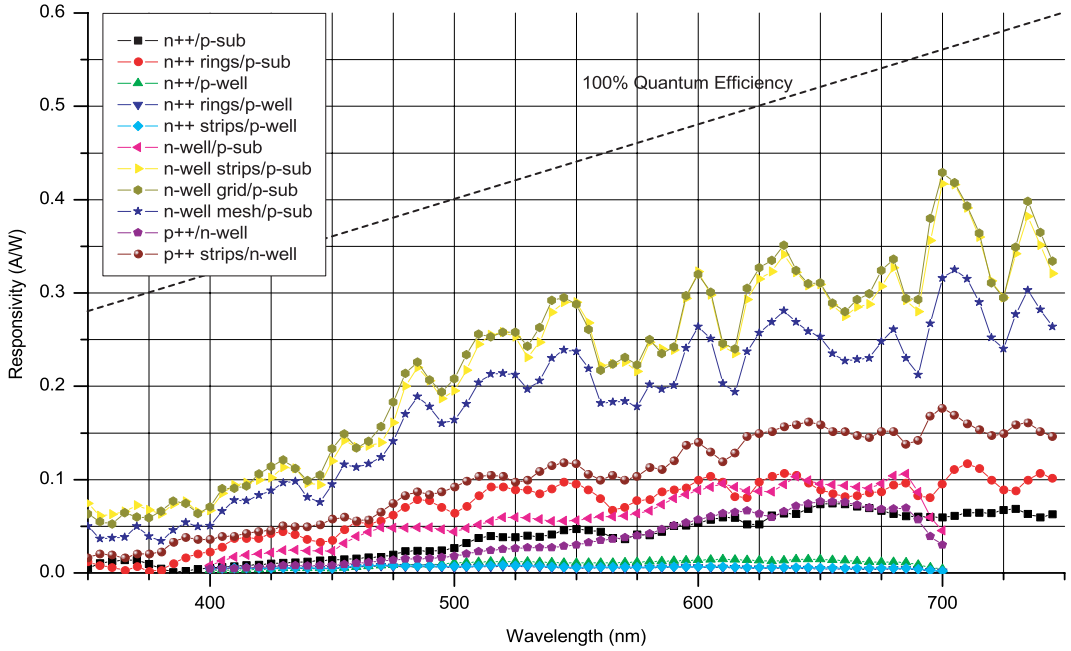


Figure 5.12: Measured spectral responsivity of single junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$).

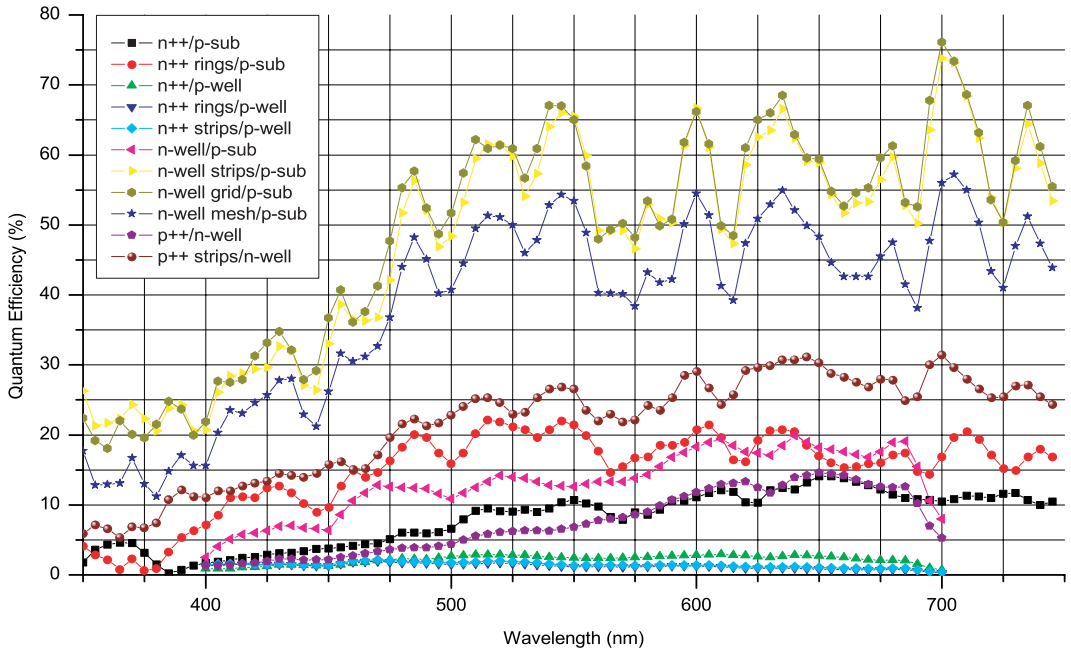


Figure 5.13: Measured spectral quantum efficiency of single junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$).

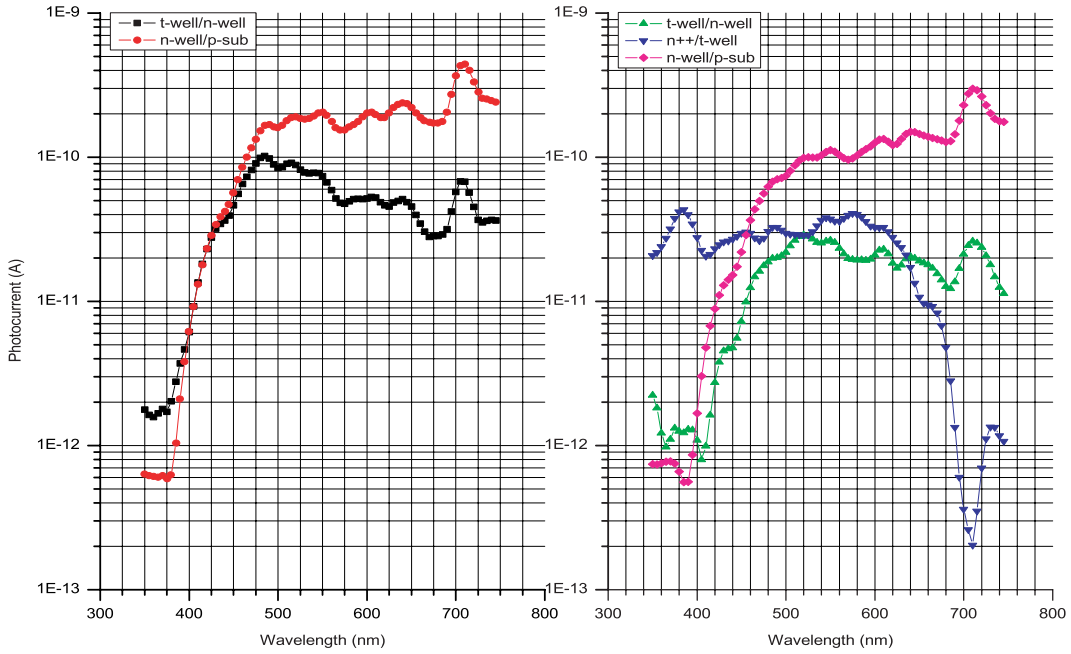


Figure 5.14: Measured spectral photoresponse of multiple junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$). Devices tested are: t-well/n-well/p-sub (left) and n++/t-well/n-well/p-sub (right).

shallow junctions ($< 0.5\mu m$) are selective to short wavelength light, i.e. blue, whereas deeper junctions ($> 1.5\mu m$) are selective to long wavelength light, i.e. red. The actual spectral selectivity is expected to be better than the presented results, as the measurements were taken separately for each junction. Therefore, stray electron-hole pairs generated at other junction interfaces may contribute to neighbouring junction spectral performance. This would have the effect of observing reduced spectral responsivity.

- Spectrally insensitive devices: The normalised quantum efficiency results for devices that do not show spectral selectivity are presented in Fig. 5.19. Generally deep devices with relatively high sidewall to base ratio are spectrally insensitive. The corresponding normalised quantum efficiency curves show similar spectral profiles for a number of such devices, exhibiting an almost flat response from $500nm$ to $700nm$ allowing for optical interference effects.

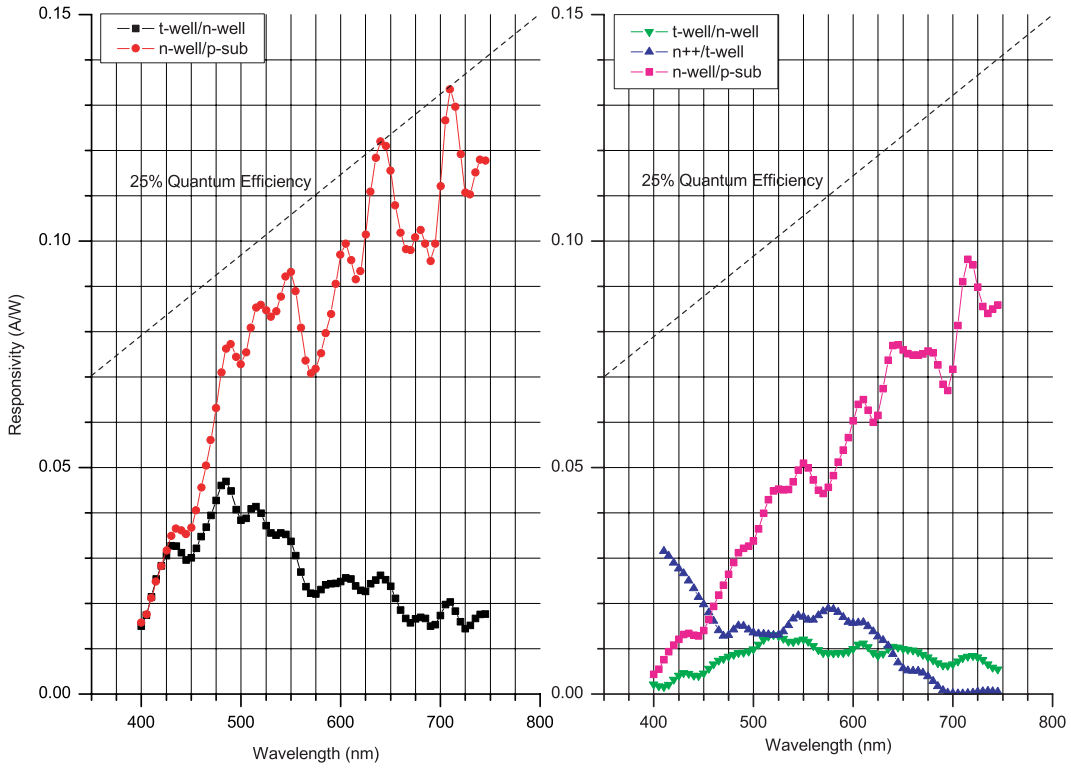


Figure 5.15: Measured spectral responsivity of multiple junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$). Devices tested are: t-well/n-well/p-sub (left) and n++/t-well/n-well/p-sub (right).

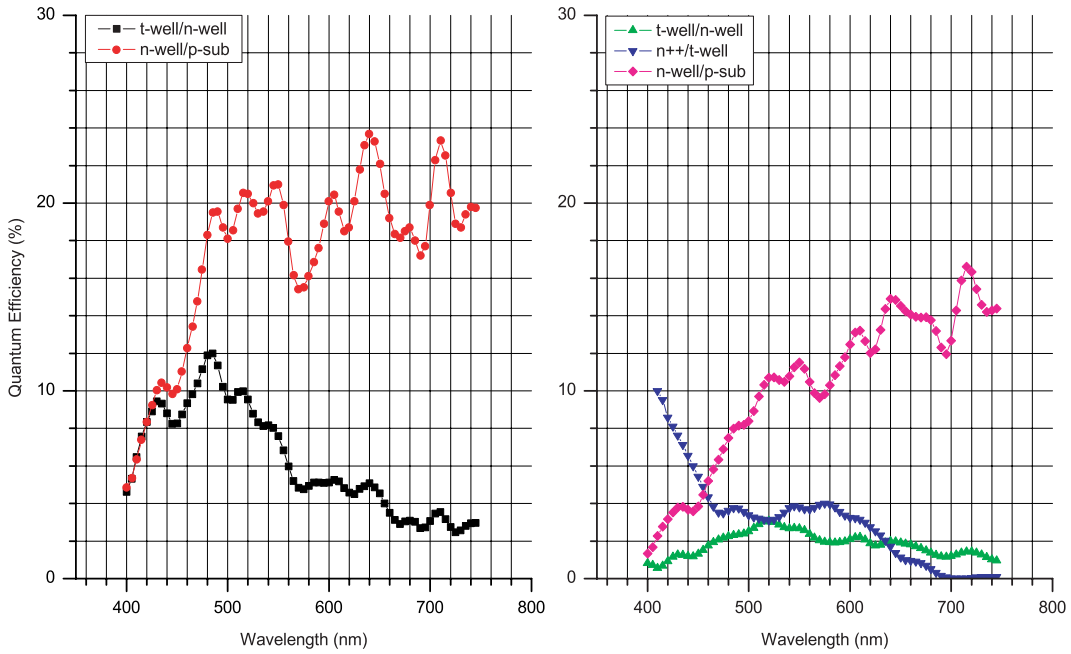


Figure 5.16: Measured spectral quantum efficiency of multiple junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$). Devices tested are: t-well/n-well/p-sub (left) and n++/t-well/n-well/p-sub (right).

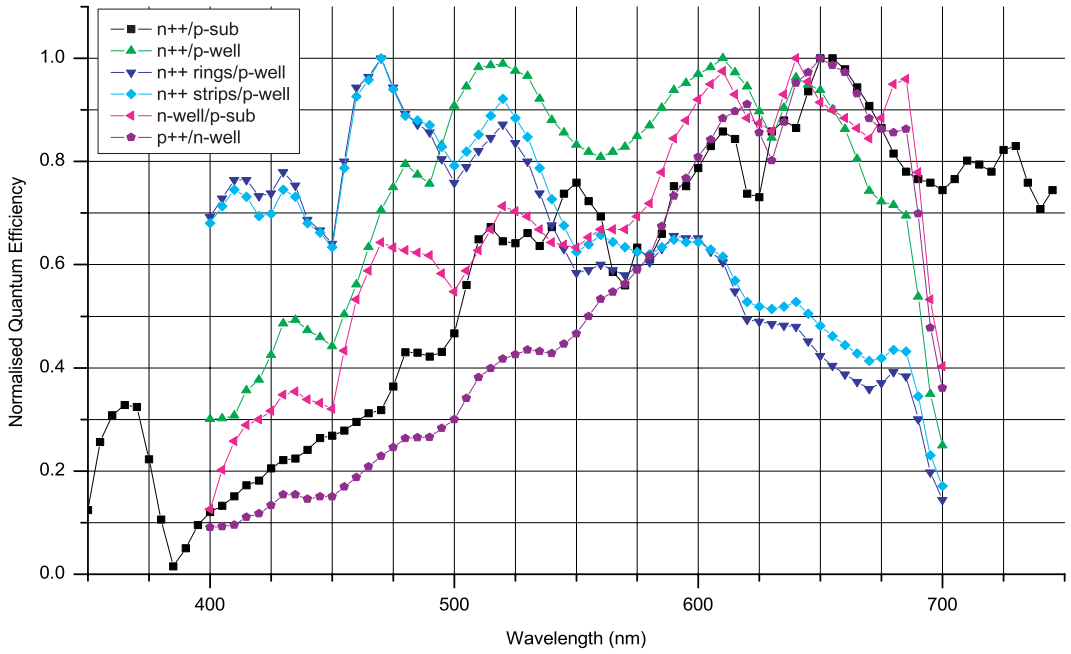


Figure 5.17: Measured spectral quantum efficiency (normalised) of spectrally selective single junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$).

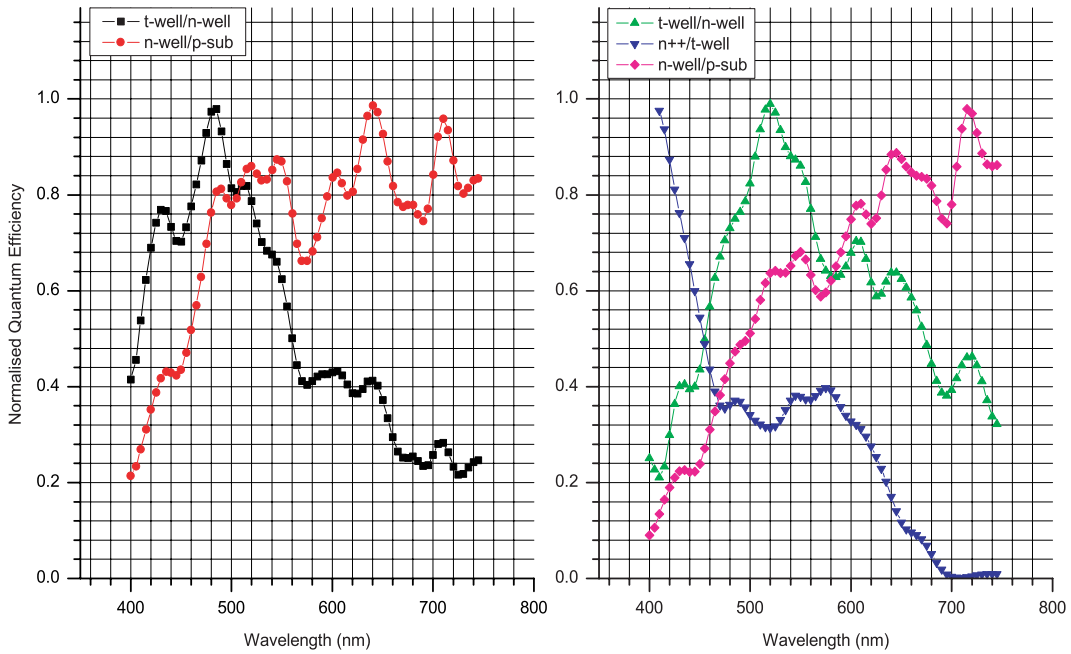


Figure 5.18: Measured spectral quantum efficiency (normalised) of multiple-junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$). Devices tested are: t-well/n-well/p-sub (left) and n++/t-well/n-well/p-sub (right).

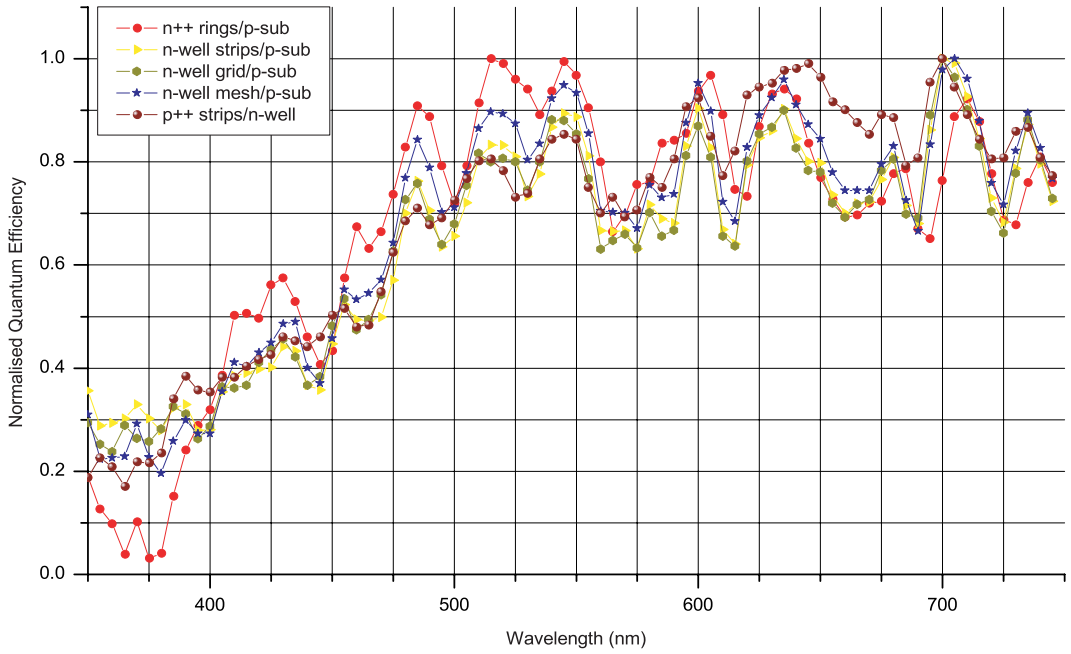


Figure 5.19: Measured spectral quantum efficiency (normalised) of spectrally unselective single junction photodiode structures (using controlled light source: $350nm < \lambda < 750nm$).

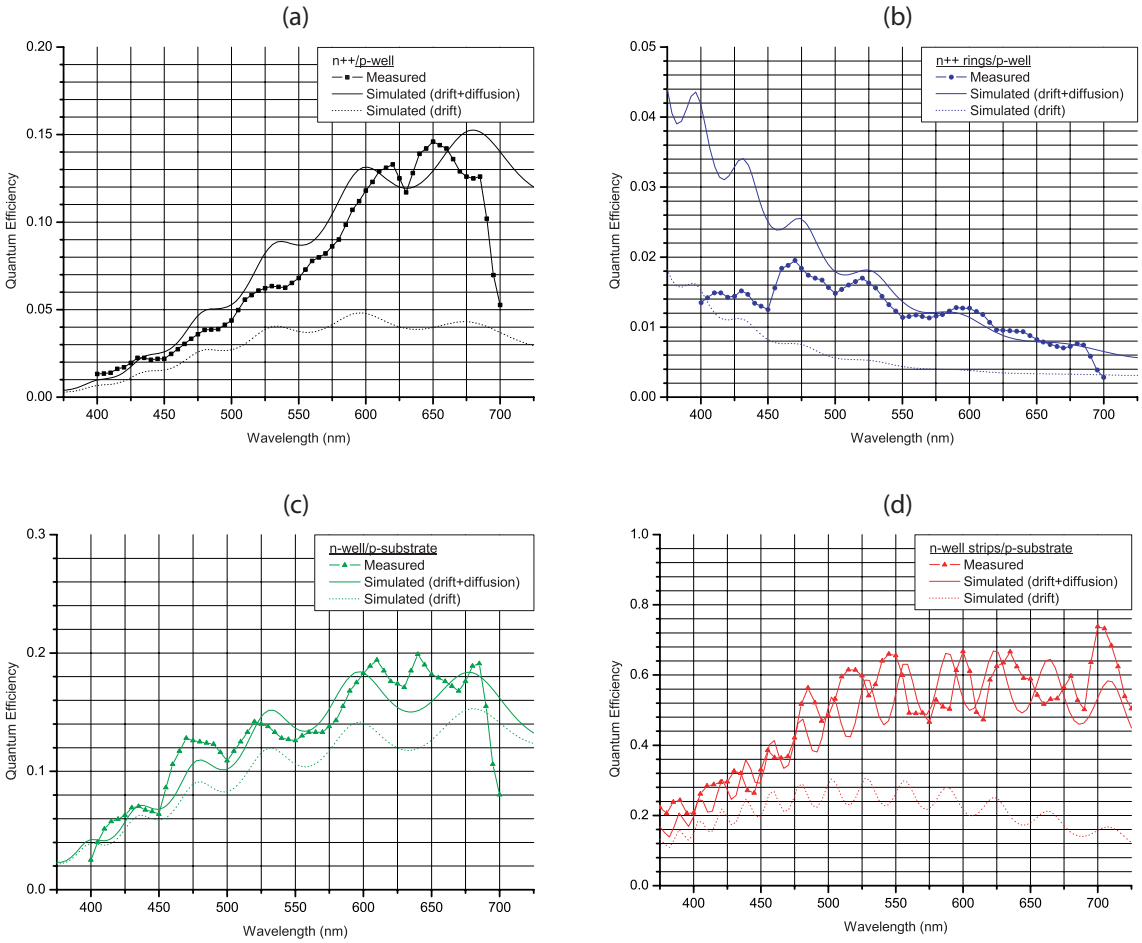


Figure 5.20: Measured and simulated (based on developed photoresponse model) spectral quantum efficiency comparison for: (a) n++/p-well, (b) n++ rings/p-well, (c) n-well/p-substrate and (d) n-well strips/p-substrate.

5.4 Photodiode Results Discussion

5.4.1 Functional Analysis

A functional analysis of experimental data has been performed using the presented device measurements and simulation results, based on the developed photodiode model. The comparisons between measured and simulated results for spectral (external) quantum efficiency are illustrated in Fig. 5.20.

The results compare measured and simulated results by initially considering only the

drift current contribution (in simulation model) and subsequently also including the diffusion contribution to provide a more realistic model. From the four examples; the devices tested being: n++/p-well, n++ rings/p-well, n-well/p-substrate and n-well strips/p-substrate, in three cases the diffusion current makes up a substantial proportion of the total photocurrent (up to 50% or more). For the n-well/p-substrate device (Fig. 5.20c), the diffusion current component constitutes only 10-15% or so of the total photoresponse; accentuated for longer wavelength light. This can be explained by the fact that this device has been designed to have no exposed (optically) sidewall region, thus there is no lateral diffusion current contributing to the photocurrent (although it contributes to the dark current). Thus the only diffusion current occurs at the vertical junction; principally below the junction in the substrate. As this is a n-well junction, the region contributing diffusion current would be located over $2\mu m$ beneath the semiconductor/coating (silicon dioxide) interface. At this depth, only light of longer wavelength (beyond 550nm) would be able to be absorbed.

Concerning spectral selectivity, the simulated results generally conform to the measured data; following a similar trend. The only exception is the n++ rings/p-well junction device (Fig. 5.20b), where measured photoresponse is approximately only 40% of the expected value for short wavelengths ($350nm$-$450nm$). However, above $450nm$ the measured and simulated results conform reasonably well. This discrepancy can be attributed to recombination just below the semiconductor/coating (silicon dioxide) interface due to surface effects. The reason this appears accentuated is because this device is designed with very shallow junctions and a very high lateral to vertical area ratio; thus virtually all photoresponse is expected to be due to lateral diffusion near the surface. One might expect the n++/p-well device (Fig. 5.20a) to suffer from similar effects, however this is not the case as it contains a single junction with no exposed (optically) lateral junctions. Subsequently most the diffusion current is below the junction into the well and therefore has no direct route to the semiconductor surface.

The measured and simulated spectral profiles reveal the optical interference effect is more intricate than expected. The model includes a single air/coating/substrate interface and determines the interference effect due to the coating thickness; considered a single interface. However, in practice the optic coupling between air and substrate involves many embedded dielectric layers; evident from the interconnect cross section (Fig. 5.3). Subsequently, although primary effects have been successfully modelled, the measured results tend to suggest that there are additional secondary processes further degrading the inci-

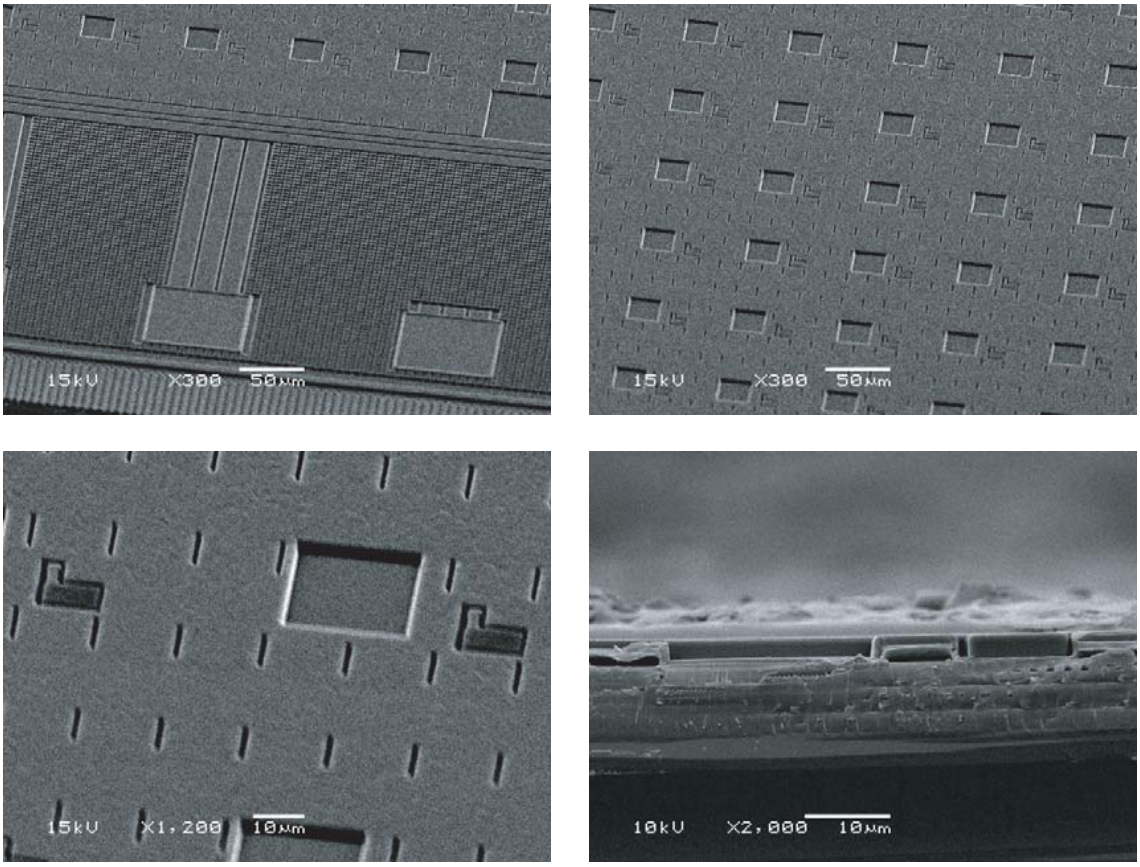


Figure 5.21: Scanning electron microscope (SEM) images of the ORASIS-P2 surface; illustrating the passivation layer/air interface profile. Cross-section through a photodiode region shown in lower-right image.

dent light radiation. These are caused by internal absorption, reflections, refraction and thus scattering within the layered coating; created during the planarisation process.

Furthermore, the actual fabricated device surface profile substantially deviates from the process data over the photodiode regions. This is due to these regions violating metal fill constraints required for uniform planarisation. As a result, the surface becomes slightly indented with pit-like features at the photodiode openings. This can be clearly seen from the electron microscope images, shown in Fig. 5.21. From these images the indentation is measured to be $2.5\mu m$.

5.4.2 Impact of Technology Scaling on Photodiode Performance

As CMOS technology inevitably progresses and scales, it has many detrimental effects on CMOS technology being used in applications requiring photodetection and in particular imaging. Technology scaling affects imaging devices in the following areas:

- Sensitivity: Higher doping concentrations lead to reduced mobility and carrier lifetimes. As a result the diffusion length is reduced thus reducing the diffusion current contribution to the overall photoresponse. Furthermore, higher doping leads to reduction in depletion layer width; thus also decreasing the drift current contribution to the overall photoresponse.
- Dark Current: Reduced diffusion length results in increased dark current density thus further degrading the signal to noise ratio (SNR).
- Dynamic Range: Technology scaling results in reduced gate oxide thickness and thus also reduced power supply voltage. For Active Pixel Sensor (APS) applications, this means a reduction in maximum signal level, thus further limiting the overall dynamic range.
- Spectral Response: Shallow drain/source diffusion provide structures capable of absorbing short wavelength light. As a result, spectral sensitivity tends to shift to shorter wavelengths with technology scaling.
- Optical Interface: Increasing interconnect layers means more oxide surfaces and increased thickness, therefore more internal interfaces thus increased interference effects. Reflection, refraction, diffraction and absorption cause scattering and attenuation leading to reduced device efficiencies and increased inter-pixel cross-talk.
- Junction Capacitance: Increase doping results in reduced depletion region widths and therefore increased junction capacitance.

5.4.3 Photodiode Design Recommendations

- For high device efficiencies:
  - Design junction structures using minimally doped structures, typically substrate/well diodes.

  - Use several small parallel-connected structures, i.e. to achieve high lateral to vertical area ratio.
  - Distribute substrate contacts throughout a multiple junction devices to maximise collection efficiency.

- For increased optical efficiency:
  - Select technology and/or option with minimum required metal layers.
  - Select technology with anti-reflective coating (ARC) or post-process.

- For colour selectivity use single diode structures (vertical junction), optically shielding the sidewalls (lateral junctions).

- Position metal interconnects near device perimeter to minimise shadowing and/or diffraction effects.

- Use high reverse bias for minimal capacitance.

- To minimise cross-talk:
  - Use deep guard ring, typically a biased well if area permits, otherwise maximise substrate contacts around the perimeter.
  - Include perimeters throughout all metal layers connecting via interconnects, forming a cage structure.

5.5 Interface Techniques

Interfacing to a photodiode is perhaps the most critical and important circuit within a vision chip. Selecting the correct topology is crucial to overall system performance and success. The most popular techniques are illustrated in Fig. 5.22.

5.5.1 Continuous-time Pixel

The simplest circuit and most popular (throughout the vision chip community) for converting a photocurrent into a voltage is the logarithmic sensor (See Fig. 5.22a,b) using a stacked diode-connected MOS devices [15] biased in weak inversion by the photocurrent itself.

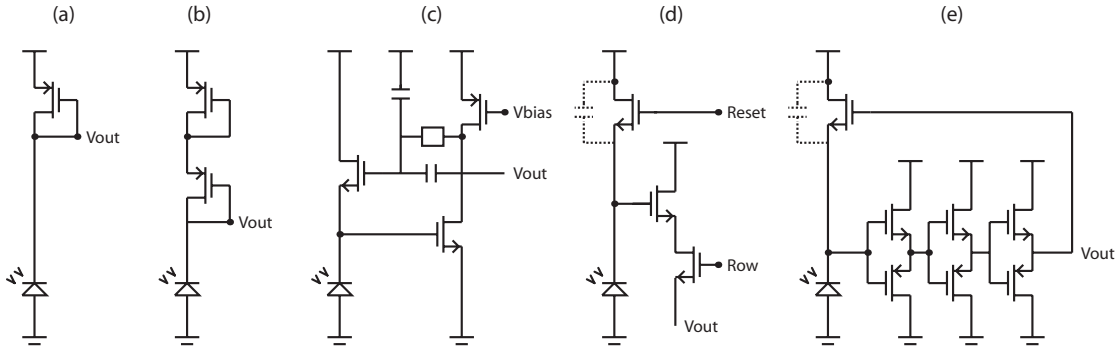


Figure 5.22: Various photodiode interface topologies. Shown are: (a) logarithmic sensor using single MOS diode (b) logarithmic sensor using two series MOS diodes (c) adaptive photoreceptor (d) active pixel sensor (APS) and (e) spiking photoreceptor

- Advantages: Small size (silicon area) and wide dynamic range.
- Disadvantages: High sensitivity to device mismatch and slow response in low light conditions.

The continuous-time logarithmic sensor has been widely used throughout the vision chip community. An popular variant of a continuous-time logarithmic pixel has been the adaptive photoreceptor circuit (See Fig. 5.22c) [16]. This has incorporated a logarithmic photoreceptor topology with a temporally adapting bias, thus altering its operating point over time and achieving an impressive dynamic range.

5.5.2 Active Pixel Sensor (APS)

The photodiode interface topology used in all CMOS imaging systems is the APS organisation (see Fig. 5.22d) [17]. The photocurrent is used to charge up a parasitic MOS capacitance that is periodically sampled and reset. This technique has several advantages: linear transfer, controllable dynamic range and low sensitivity to device mismatch. The main disadvantages are: the dynamic range cannot be set locally and it requires a clock.

- Advantages: Linear transfer characteristic, controllable dynamic range and low sensitivity to device mismatch.
- Disadvantages: Global clock required and local adjustment of dynamic range has not been achievable until recently [18, 19].

5.5.3 Spiking Pixel

An alternative technique is to output the result as a frequency (or pulse/spike rate). The implementation is similar to the APS, except that the photocircuit self-resets itself (see Fig. 5.22e) [20, 21]. By monitoring the integrating node by means of a comparator, the reset switch can be activated after a set threshold has been surpassed.

- Advantages: Information encoded temporally (continuous time, discrete signal) therefore robust to device mismatch and noise pickup. Asynchronous technique; requires no clock.
- Disadvantages: This method intrinsically has a slow response in low light conditions. Recent developments to overcome this use a time-varying threshold [22] or using ON/OFF encoding [23, 24]. Furthermore a spiking output generally requires higher communication bandwidth, however, recent work has attempted to address this by using single spike coding

5.6 An Adaptive-ON/OFF Spiking Photoreceptor

In this section is presented a spiking photoreceptor circuit [24] which is intended for use in adaptable foveating vision chips. The ultimate aim is to realise an imaging device which can electronically split its photosensor array into peripheral and fovea regions in a similar fashion to the human eye. Specialisation of the visual field would allow for high spatial or high temporal resolution imaging with the possibility of high dynamic range. To this end a pulse frequency modulated spiking photoreceptor has been developed which is capable of providing high dynamic ranges with power consumption similar to animal retina. The circuit is based on the ON/OFF opponency algorithm used by the human eye to maintain high frequency response at low light levels, while maintaining low power operation. The photosensor surface fill factor is kept to a maximum and power consumption are kept to a minimum. This section discusses the algorithm, its implementation and simulated/measured results describing its response and power consumption.

5.6.1 A bio-inspired encoding scheme

The eye has around 100 million photoreceptors with an intensity detection range from starlight to bright sunlight, and a typical video rate of 25 Hz. This remarkable capability is rooted in the rhodopsin photocascade [25] in the rods and cones which are used to convert incoming photons into electrical information.

Inorganic silicon photodiodes are capable of up to five orders of magnitude of dynamic range. Usually however, only an 8-bit dynamic range per channel is implemented due to the relatively high power consumption of higher bit-rate signal conditioning circuits. This can under or over saturate scenes with large variations in image intensity. Early work by Delbrck and Mead [16] led to an adaptive photoreceptor which could detect the contrast regardless of overall light intensity. This structure along with edge detection algorithms [26, 27] can be used to extract the salient information from the scene, at the expense of the non-salient information.

It is however possible to use a spike rate encoding algorithm similar to that in animal vision such as the human eye [28, 29]. By changing from voltage or current space to frequency space, it is possible to achieve wide dynamic ranges at lower power consumption. Furthermore, such a scheme can provide adaptive functionality, trading between high dynamic range and frequency response by means of adaptive spike counting.

The major drawback to any integrating system is the low frequency response for low light intensities. Here again we can learn from nature by implementing complementary ON and OFF channels [23, 30]. Using ON/OFF opponency, where ON-cells spike at high frequency at high light levels, and OFF-cells spike at high frequency at low light levels, there will always be an adequate frequency response, even at low light levels. Therefore either the ON- or OFF-cell will always provide a high firing rate and thus a fast frequency response. A winner-takes-all type of circuit can be used to remove redundancy in the output information stream.

A naive approach to the total spiking rate, before compression could be expected to be given by:

$$f_{spike} = DR \times N_{photoreceptors} \times f_{refresh} \tag{5.26}$$

Where f_{spike} is the output spiking rate, DR is the dynamic range, $N_{photoreceptors}$ is the total number of photoreceptors range and $f_{refresh}$ is the refresh rate.

Therefore, assuming a dynamic range of 7 decades, 100x106 photoreceptors, and a refresh rate of 25Hz, the output spiking frequency would be an enormous 2×10^{16} spikes per second. Even with the 5 million or so axons; constituting the optic nerve, leading to the visual cortex, this would be unobtainable.

The retina therefore carries out various algorithms to sort salient information from non-salient information, and to compress information stream to the visual cortex. This processing includes spatiotemporal filters [27], colour opponency and motion sensitivity. The retina also specialises into the fovea and peripheral vision. The fovea contains a high concentration of cones and is scanned across the field of view to build up a high definition image, whereas the peripheral vision concentrates on passing on fast motion information and is important for object fixation. The dynamic sensitivity in intensity is compressed using the iris as a light intensity modulator. The division into regions of fast motion sensitivity and high spatial sensitivity has been very successful in evolutionary terms. Most vertebrates perform this type of processing to get round bandwidth restraints in sending visual signals to the visual cortex. The fast temporal resolution is important for danger awareness, and reaction time, while the spatial resolution allows greater understanding of ones environment.

Therefore, in developing such a biologically-inspired system, a technique is required for dynamically changing the output of an imaging device from high temporal but low spatial resolutions to low temporal but high spatial resolutions.

5.6.2 Photodiode Implementation

The measured current/intensity characteristics for the photodiode to be used can be seen in Fig. 5.23. As previously seen, generally the psub/nwell or psub/n+ structures are most efficient. Internal quantum efficiency in these structures tend to be high. However external quantum efficiencies tend to be lower due to fill factor constraints coupled with coupling losses due to the dielectric layers and the surface morphology of the CMOS chip around the photodiode.

The response is linear for an incident power from $50pW$ to greater than $100nW$. The minimum detectable light is set by the dark current which is determined by the bias and the fixed pattern noise. While increasing bias increases the frequency response and quantum

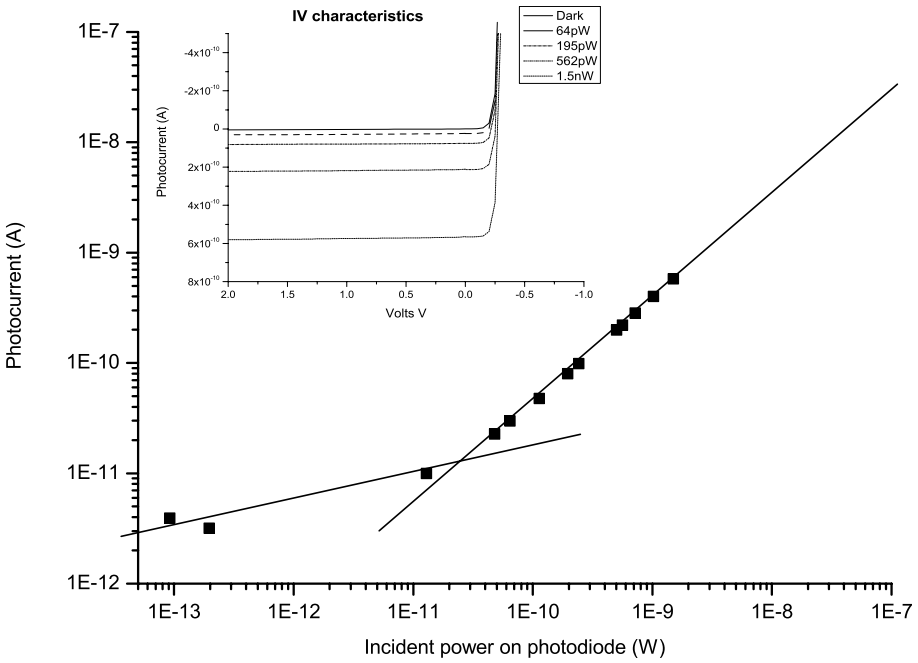


Figure 5.23: Measured photodiode characteristics. The photodiode response is linear until below 50pW.

efficiency, it also increases the dark current by a much greater factor. In our configuration the photodiode is reverse biased by 1.5V leading to a dark current of $4.8fA\mu m^{-2}$. Fixed pattern noise is also a limiting factor as it can reach 1% (between pixels at $100\mu m$ proximity) on the amplification stage, but as will be discussed later, our asynchronous spiking regime allows for some variance as we will discuss later. The photodiode internal quantum efficiency is 76% for $530nm$ wavelength.

The current characteristics from this photodiode were used in the circuit simulation for the spike generator. For the purposes of the circuit simulation we have used a current variation of $1pA$ to $10nA$ corresponding to $25nWcm^{-2}$ to $2.6mWcm^{-2}$. These five decades of intensity variation correspond to the difference between starlight and a well lit room. The photocurrents on a seven pixel photoreceptor group can be added to give better dynamic range in dark conditions. This can be seen in the seen in the system algorithm given in Fig. 5.24.

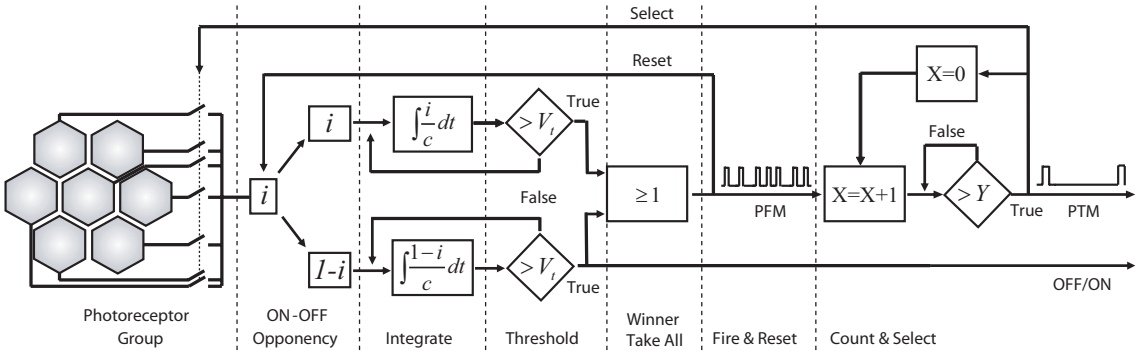


Figure 5.24: Core algorithm per photoreceptor group includes: phototransduction, ON/OFF opponency, spike generation, variable spike interval encoding and input selection.

5.6.3 System Algorithm

In previous neuromorphic vision chips pixel sizes have tended to be around a 100x100μm in size with a photodiode fill factor of around 10% [31]. This has tended to work against creating imaging chips with high pixel densities. High pixel density therefore requires effective processing that does not take up large areas of the imaging array.

The basic system algorithm can be seen in Fig. 5.24. A single spike encoder is shared between seven photodiodes, thus increasing the effective fill factor. The spike encoder can take inputs from individual or all of the photodiodes using a switched arrangement. This provides the system the ability to select between high spatial and high temporal resolution, and decreases the silicon surface area required. The spike encoder takes the selected photocurrent(s) and produces complimentary increasing and decreasing currents to provide the ON and OFF channels, i.e. low photocurrents create high OFF-currents and vice versa. The two channels then compete by integrating their currents into voltages through capacitance. This voltage is released in the form of a spike once a trigger threshold has been surpassed and the charge collected is reset. To reduce redundancy only the first spike, whether ON or OFF is released and both channels are reset. A complementary output is sent to indicate an ON or OFF spike. Hysteresis is added to stop the circuit oscillating between on and off when the light intensity is close to the threshold between light and dark channels. This relatively high spiking response is then compressed by means of a counter. On overflow, a spike interval encoded (SIE) output is signalled, resetting the counter and selecting the next photodiode in sequence. The counting method enables reconfigurability

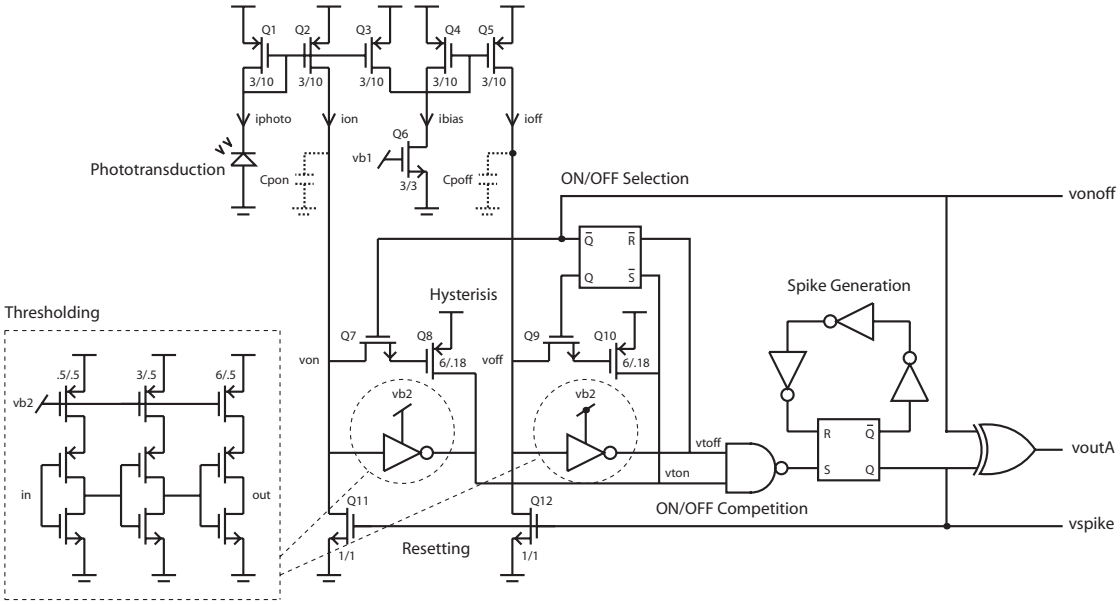


Figure 5.25: Circuit schematic of the adaptive-ON/OFF spiking photoreceptor block, operated off a 1.8v core supply (implemented in 0.18m CMOS). Illustrated is the basic scheme for generating an adaptive-ON/OFF spiking output for a single photodiode input. Shown at the bottom-left is the implementation of the thresholding comparator based on a scaled cascade of current-limited digital inverters. Unless stated all devices have aspect ratio (4/3)lmin for NMOS and (10/3)lmin for PMOS, with lmin being the technology minimum feature size.

and tradability between dynamic range and temporal response by means of selecting the overflow at different stages.

5.6.4 Circuit Topology and analysis

The circuit schematic is given in Fig. 5.25. The pn photodiode is reverse-biased by the diode-connected PMOS device Q1; forming a simple current mirror with devices Q2 and Q3. To ensure good linearity, devices Q1-Q5 are sized such that they operate being biased in weak inversion over the full input current (photocurrent) range. The photodiode reverse-bias voltage is therefore given by:

$$V_{photo} = V_{GS1} = nV_T ln(I_{photo}/I_0) \tag{5.27}$$

Where n is the slope factor, V_T is the thermal voltage ($= kT/q$) and I_0 is the device pre-exponential current. For photocurrents of 100fA to 1nA, the photodiode reverse-bias is: $1.47 \leq V_{photo} \leq 1.78$ for $V_{dd} = 1.80V$. Subsequently I_{photo} is mirrored by devices Q2 and Q3 to source the ON-current (I_{on}) and ON-difference-current (I_{on}). The ON-difference-current is generated by means of Kirchhoff's current law (KCL), i.e. by injecting the copied ON-current (I_{D3}) into a current sink (I_{D6}), the difference can be determined. By sourcing this difference through a PMOS current mirror, the OFF-current ($I_{ioff} = I_{D5}$) is generated.

$$I_{off} = I_{D5} \approx I_{bias} - I_{photo} \tag{5.28}$$

The ON and OFF currents are then used to create an increasing voltage, by means of integrating these into the parasitic capacitance of their respective nodes. These capacitances are given by:

$$Cp_{on} = C_{DB2} + C_{GD2} + 3(C_{DB7} + C_{GD7}) + C_{DB11} + C_{GD11} \approx 1.3(C_{DB2} + C_{GD2}) \tag{5.29}$$

$$Cp_{off} = C_{DB5} + C_{GD5} + 3(C_{DB9} + C_{GD9}) + C_{DB12} + C_{GD12} \approx 1.3(C_{DB5} + C_{GD5}) \tag{5.30}$$

Where the predominant capacitance is due to the current sourcing devices (Q2, Q5) and the reset switches (Q11, Q12). Therefore selecting reduced device widths for these devices can reduce this capacitance for higher speed operation. The limiting factor to how much these device widths can be reduced to depends on device matching for Q2, Q5 and $R_{DS}(on)$ for Q11, Q12. For the designed values, i.e. W/L(Q2,Q5)=$(3/10)\mu$ and W/L(Q11,Q12)=$(1/1)\mu$, this node capacitance is 7.2fF. Therefore, the maximum spiking rate is given by:

$$f_{max} = \frac{I_{bias}}{Cp_{off} V_{threshold}} \tag{5.31}$$

Where $I_{bias} = I_{photo}(max)$ is the maximum ON or OFF current and $V_{threshold}$ is the comparator threshold voltage. The threshold comparators (shown in Fig. 5.25) are based on a high-gain digital inverter cascade. By limiting and scaling the maximum current source to each stage, a successively steeper edge is obtained and power consumption is minimized.

For a three-stage cascade of minimum channel length devices, the optimum (for power consumption) first stage bias current is 2nA with a current ratio of 1:6:12 between stages. This gives a comparator threshold voltage of 270mV on a 1.8V supply. Therefore using expression Eqn. 5.31 the maximum frequency of operation for a 1nA maximum photocurrent is: fmax=(1n)/(7.2f)(270m)=514.4kHz.

By threshold detecting the integrating nodes V_{on} and V_{off}, the first channel (ON or OFF) to reach threshold is collected through a logic OR operation. This provides the V_{spike} output. A minimum spike width is then secured by using a digital monostable based on a self-resetting RS flip-flop with inverter cascade. This is required to ensure a minimum pulse width is asserted on the reset switched to reliably discharge the integrating nodes each integration period. An additional output is provided to specify whether the response is ON or OFF by using an additional RS flip-flop to determine which channel is dominant. Hysteretic feedback is provided to the threshold comparators by slightly increasing the threshold voltage by 5mV providing an approximately 10% lag on channel selection changeover to prevent rapid channel toggling when the ON and OFF responses are comparable. The two outputs V_{spike} and $V_{ON/OFF}$ are also combined to provide a single spike polarity encoded (SPE) signal V_{SPE} by using a logic XOR operation.

5.6.5 Circuit Implementation

The adaptive-ON/OFF spiking photoreceptor circuit has been designed, implemented and fabricated in a standard 0.18μm CMOS process. The single receptor layout is shown in Fig. 5.26, the total silicon area being 880μm^2. This would lead to a fill factor of 52% with 30μm x 30μm photodiodes. However, it is possible to share the spike generator circuit amongst multiple photodiodes to increase the fill factor and/or photodiode density. In our configuration we envisage sharing the photoreceptor between a local hexagonal neighbourhood containing seven photodiodes, as given by Fig. 5.24.

5.6.6 Simulated and Measured Results

The circuit was simulated using the Cadence Spectre (5.0.33) simulator with BSIM 3v3.2 models for the MOS devices combined with a photodiode model derived from the measured parameters, shown in Fig. 5.23. The simulation results for the individual ON and OFF channels are shown in Fig. 5.27. The slow responses can be seen for the ON channel at low

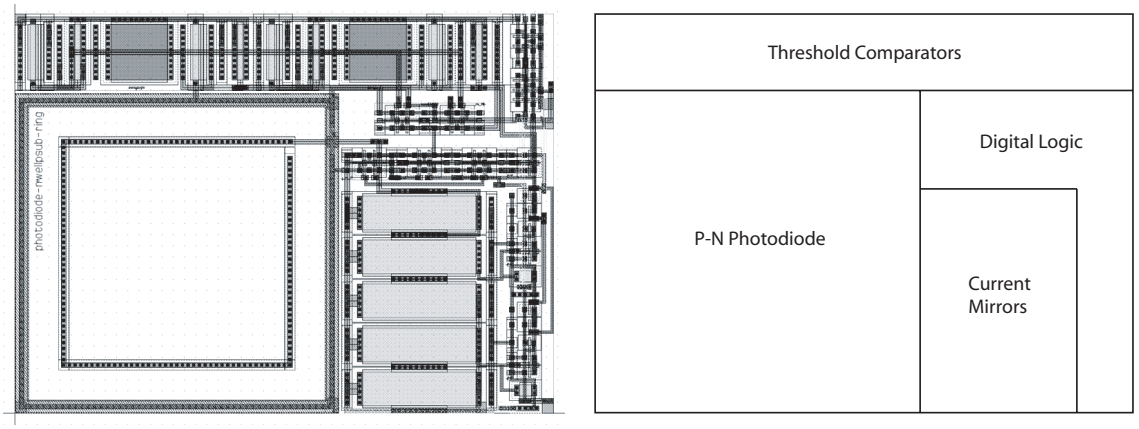


Figure 5.26: Physical layout of the adaptive-ON/OFF spiking photoreceptor block, implemented in UMC 0.18μm Mixed-mode CMOS. By area, the photodiode has a 52% fill factor, the threshold detectors occupy 18%, current mirrors 16% and asynchronous digital logic 14%.

light intensities and the OFF channel at high light intensities. The simulation results of the competing ON/OFF channel spike generator are shown in Fig. 5.28. The response shows good variation between light and dark over many orders of magnitude and the final output shows good distinction between ON and OFF channels. The hysteresis at the transitions can also be clearly seen.

The spike interval at the maximum firing rate is 2μs, corresponding to over 1MHz in response when considering that the data is effectively compressed by half. 500kHz is sufficient to provide a 16-bit dynamic range at 5Hz refresh on a single pixel. As mentioned previously, this maximum firing rate is limited by the combined parasitic capacitance at the integrating nodes as described by expressions 5.29, 5.30 and 5.31.

The power dissipation of this circuit can be expressed due to two sources; the continuous current flow in the current mirrors; the static power and the digital switching; the dynamic power. The total current consumption is illustrated in Fig. 5.29. The quiescent (or static) current consumption is approximately $3nA$ (dependant on bias, i.e. $I_{static} \approx 2I_{bias}$). The dynamic current consumption is $370\mu A$ per $1.5ns$ spike in a $3\mu s$ window. Thus the energy consumption per spike is $500fJ$. Given the competition between the ON and OFF channels the minimum frequency the circuit will operate at is $500Hz$. In this regime the quiescent power consumption is $5nW$ compared to $125pW$ for the spiking. However for most of the operation at $5kHz$ to $500kHz$ it is the quiescent power consumption which will dominate. Thus, averaging this quiescent power over the pulse train gives $20.5pJ$ of energy per spike,

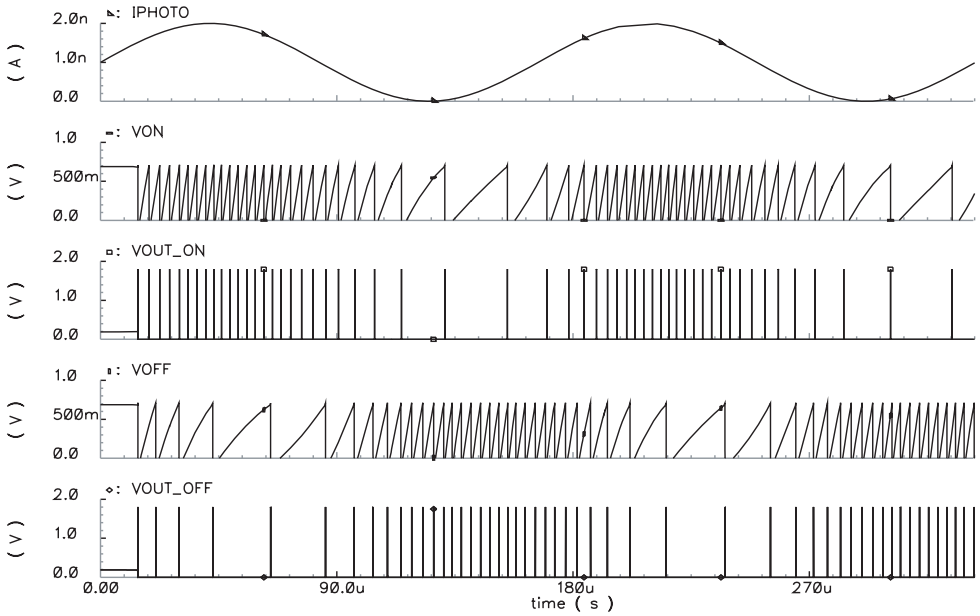


Figure 5.27: Simulated transient analysis for the individual ON and OFF channel spike generators. The waveforms shown- from top to bottom: (a) photocurrent (b) ON channel charging response (c) ON channel spike output (d) OFF channel charging response (e) OFF channel spike output.

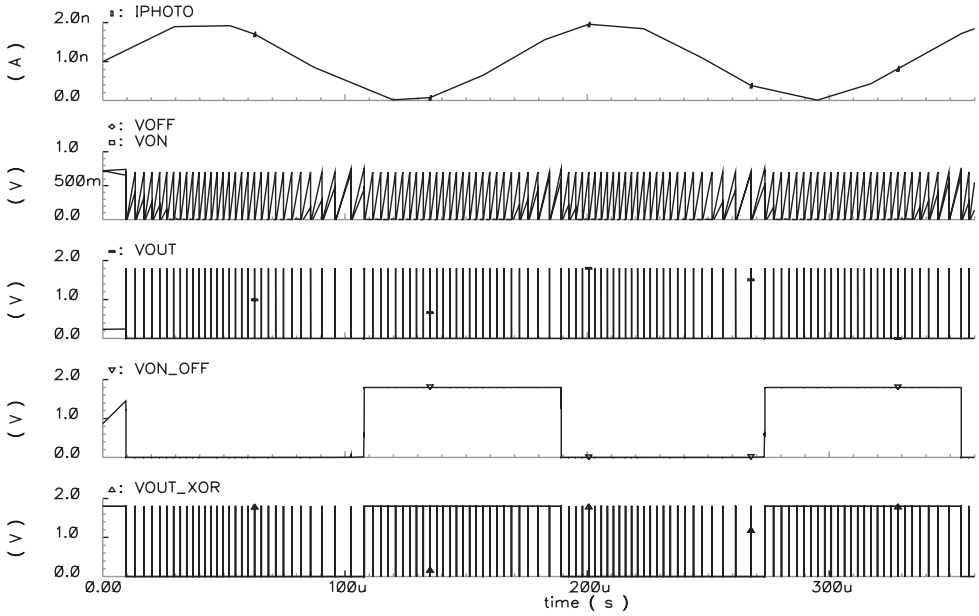


Figure 5.28: Simulated transient analysis for the combined ON/OFF channel spike generator. The waveforms shown- from top to bottom: (a) photocurrent (b) competing ON/OFF charging response (c) spike output (d) ON/OFF channel selection (e) combined spike and ON/OFF channel encoded output.

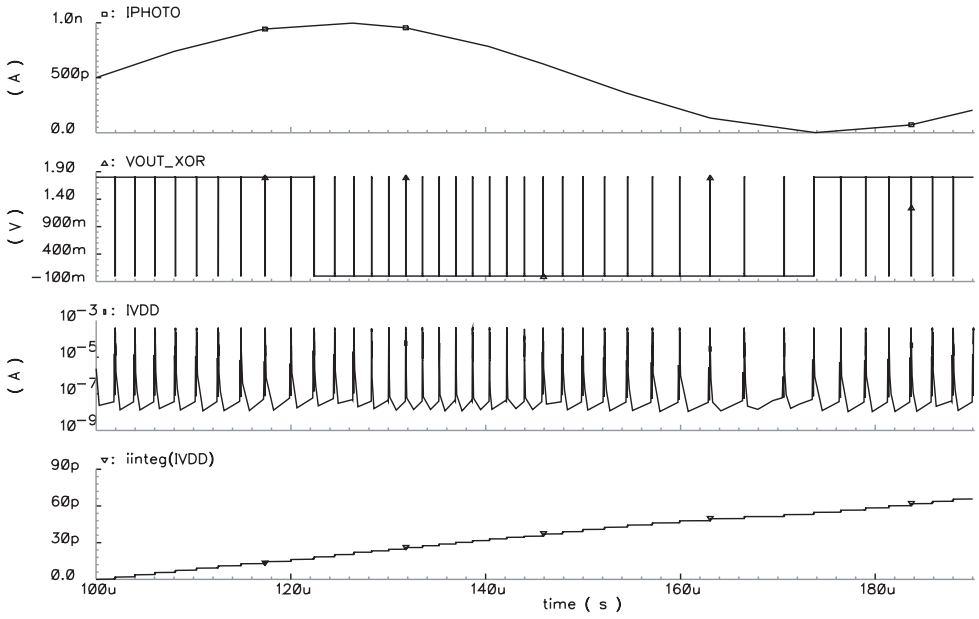


Figure 5.29: Simulated transient analysis for illustrating power consumption profile. The waveforms shown- from top to bottom: (a) photocurrent (b) combined spike and ON/OFF channel encoded output (c) current consumption and (d) integrated current consumption.

which is comparable to the bit-energy of $2 - 20pJ/bit$ for the blow fly retina [25]. An 8-bit output therefore has a power equivalent of $5nW$ per pixel.

The fabricated adaptive-ON/OFF spiking photoreceptor circuit operates as expected. The light intensity controlled frequency modulation can be clearly seen in the measured results given in Fig. 5.30.

This relationship between light intensity and spiking rate has been measured for various values of bias current, ranging from 1pA to 5nA. The response indicates there are two linear regions of operation, the boundary condition being at a 300pA bias current. This in fact agrees with the trend shown in the measured photo response of the individual photodiode, shown previously in Fig. 5.23. Furthermore, the ON/OFF response is illustrated by a positive or negative gradient in this relationship, i.e. the changeover points at the corners of the graphs. It is observed that this ON/OFF changeover point can be tuned by adjusting the bias current as would be expected. In reality only bias currents in the range $500pA$ to $5nA$ would be used to utilise the ON/OFF compression most efficiently, i.e. ideally the changeover point should be tuned to lie in the centre of input light intensity range. A high-frequency OFF response can be traded with bias current and therefore power consumption.

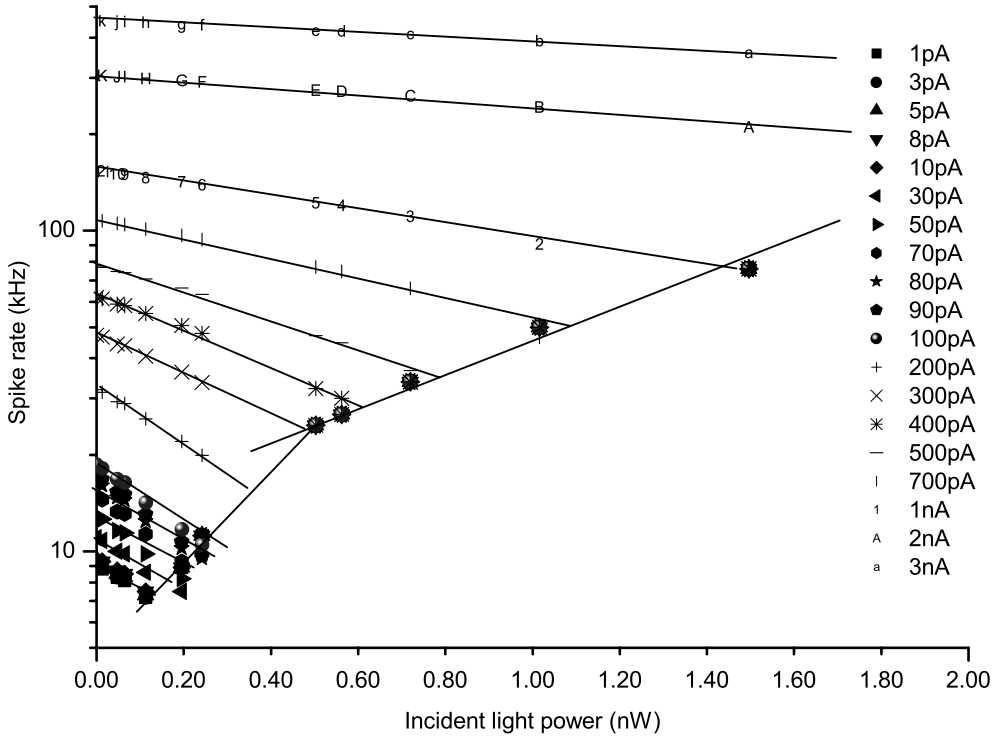


Figure 5.30: Measured photo-response results for the adaptive-ON/OFF spiking photoreceptor circuit. Illustrated is the spike rate to incident light power relationship for various bias current levels. The action to shift the ON/OFF transition point can be clearly seen. The light intensity incident on the chip is the equivalent of a well lit room.

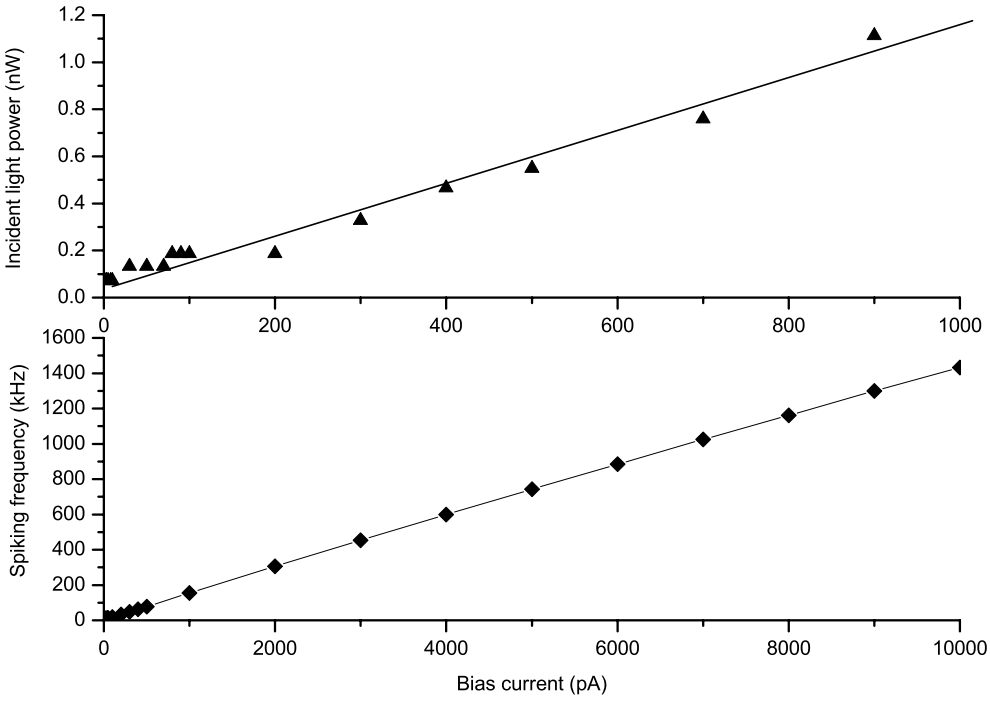


Figure 5.31: Measured bias current tuning results for the adaptive-ON/OFF spiking photoreceptor circuit. Shown (from top to bottom) is: (a) the incident light power ON/OFF crossover point versus bias current and (b) the spike rate versus bias current for dark current, i.e. zero incident light power.

The ON/OFF changeover intensity relationship with bias current is given in Fig. 5.31(a). The observed deviation from linear fit is due to two reasons in measurement procedure. Firstly due to hysteresis, an increasing intensity changeover point would be different to a decreasing intensity changeover and this has not taken into account in the measured results. Furthermore, due to the limited number of ND filters (12) used in defining the intensity variation, the changeover is measured to occur within a range rather than at an absolute value. Subsequently as the changeover for a set bias is defined by a range of two ND filter values, the error margin is quite substantial. Thus to measure a more accurate relationship either more ND filters are required, or an alternative method for intensity variation.

The dark current spiking rate versus bias current relationship is given in Fig. 5.31(b). As expected, this gives a perfect linear relationship for bias currents in the range of 100fA to 10nA. Furthermore, the fact that only an OFF response is measured for bias currents down to 100fA tends to suggest that the dark current of the photodiode biased in this circuit is below 100fA. 100fA is much better than would be expected on the basis of the photocurrent

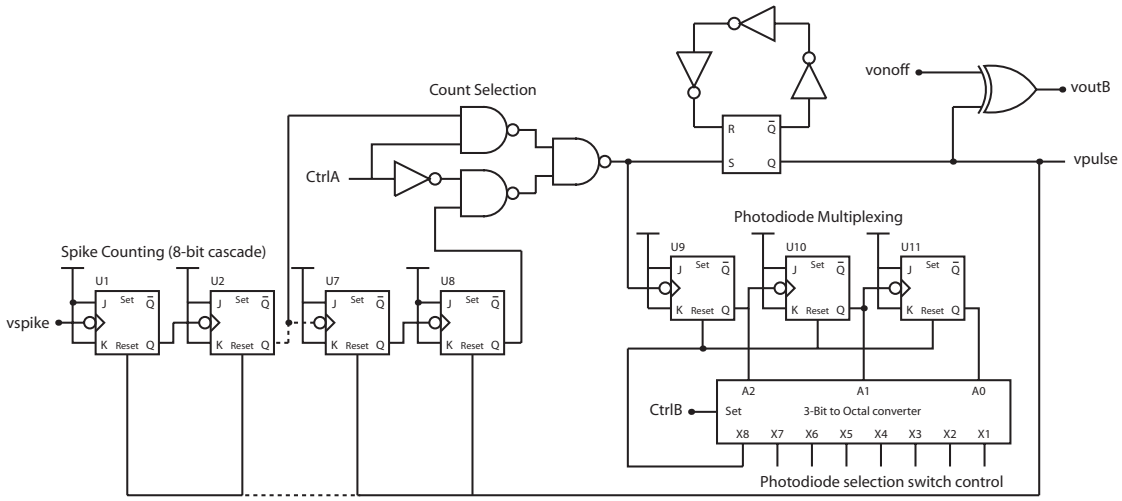


Figure 5.32: Circuit schematic of the selective output encoder/controller block, operated off a 1.8v core supply (implemented in $0.18\mu m$ CMOS). Illustrated is the spike counting circuitry for generating outputs of reconfigurable dynamic range, in addition to the photodiode multiplexing control for selection between single and multi-pixel operation.

IV curve and allows for operation in excess of 13-bits.

5.6.7 Spike Interval Encoding

The asynchronous output of the protocol would make it in the first instance very simple to address and connect to a FPGA, DSP, Address Event Protocols [32], or other digital logic. The maximum output frequency of spikes, given by Eqn. 5.26 would be $0.5MHz$. For ten sets of spike generators it would be $5MHz$ (70-140 photodiodes) and for a hundred sets of spike generators it would be $50MHz$ (700-1400 photodiodes). Clearly to achieve the equivalent of a 1 megapixel camera the maximum output frequency would reach $35-71GHz$! A 16-bit parallel bus could bring this down to a few GHz, but even this is too much for a $0.18\mu m$ process. Power consumption and switching noise would also get undesirably high. Some form of information compression is therefore required. In the eye, each spike cab convey multiple bits of information, which together with redundancy can be used to make the spiking rate more sparse [33].

The challenge is therefore to compress the data by a factor of a few decades without losing the fundamental asynchronous nature of the signal and allowing for addressing. The simplest practical way of implementing compression is to use spike interval encoded modula-

tion where each photodiodes spike train are accumulated into a single spike whose temporal width is modulated according to the spike frequency. The advantage here is multi-fold. A single temporally wide pulse introduces less switching noise than trains of spikes. Secondly the power consumption is highest on the rise and fall of the spikes. As the spiking circuit is shared between multiple photodiodes, a further advantage is that the negative pulse interval between spikes now indicates the switching to a new photodiode. Opposite polarity between ON and OFF spikes can be used to indicate their state.

This spike interval encoding can be implemented using a standard asynchronous ripple counter (see Fig. 5.32) based on a cascade of JK-type flip-flops (U1-8). Subsequently, the output can selected, using the $CtrlA$ signal from either Q(U6) or Q(U8) using a 1-bit multiplexer, i.e. divide by 64 or 256. This output encodes the compressed data and is additionally used to: (1) reset the ripple counter and (2) provide an input to the photodiode selection control circuit. This consists of a 3-bit ripple counter with an octal output for controlling the photodiode selection switches. As there are only seven photodiodes for sequential selection, the 8th output is used to reset this counter. The $CtrlB$ signal is used to foveate the local photodiode cell by overriding the multiplexer by configuring the photodiodes in parallel for increased temporal resolution.

5.6.8 Contribution to Related Work

This section highlights how the presented work compliments related work developed to similar technical goals. The contribution in this work is shown to be two-fold; both on a proposed system architecture (implementation still ongoing) and on the front-end photoreceptor circuit. At each of these levels, a brief summary on state-of-the-art related research is given followed by a rationale on how the presented work differs.

Foveating Vision Chips

Considerable work has already gone into developing vision chips based on the foveal organisation of the animal retina. Early work by Wodnicki et al. [34, 35] and Sandini et al. [36, 37, 38] produced polar arrays of increasing resolution towards the centre. More recent work by Etienne-Cummings et al. [39] has combined foveation with visual smooth pursuit tracking, acquisition saccadic control and centroiding. Azadmehr et al. [40] have produced a system with a central (static) imaging array surrounded by a temporal response

(dynamic) border for controlling a pan-tilt system to track motion on the foveal region.

The proposed system takes an alternative approach; to implement a homogenous reconfigurable array, such that the foveal region can be adjusted and moved dynamically. This provides the ability to control both the size and position of foveal region electronically, without need of mechanical actuators. Such a scheme is intended to achieve much swifter pseudo-saccadic response to an electromechanical saccade.

Spiking Photoreceptors

Spiking photoreceptor circuits have evolved since the basic concept was introduced [20, 21], not originally aimed for biologically-inspired vision chips. Along the same lines, Bermak et al. [41, 42] continued to develop a number of Pulse-Width-Modulated (PWM) and Pulse-Time-Modulated (PTM) based imagers. Another approach, due to the close resemblance to neurobiology, inspired Kramer et al. [23, 30] to use a spiking scheme to encode a temporally changing response with separate ON-increasing and OFF-increasing channels. Other developments in spiking photoreceptors included the use of current feedback to reduce energy per bit [43, 44] and using adaptive reference thresholds to achieve object segmentation [45]. To further reduce power consumption time-to-first-spike (TTFS) encoding was applied to reduce redundant spiking [46, 22, 47].

As previously mentioned (Section 5.5.3) a fundamental limitation of using spike-encoding scheme is slow response to low light levels. Qi et al. [46] addressed this issue by using a time-varying threshold, i.e. an exponential *sawtooth-like* signal. Although this technique proved to be successful in response-time, the benefits of being asynchronous and having linear response had been lost. Subsequently, this work has proposed an alternative scheme to achieve fast response whilst maintaining linear response and asynchronicity, i.e. by using an adaptable-ON/OFF encoded scheme. Furthermore, the ability to trade dynamic range with temporal and spatial resolution is made possible due to combining the proposed foveating architecture.

5.7 Summary

This chapter presents a unified model for a pn-junction photodiode implemented in CMOS technology based on the underlying semiconductor physics. Measured results of fabricated

devices have validated the quantum efficiency expressions developed. Furthermore through design, fabrication and verification, several devices have been characterised in a deep submicron process. Subsequently, some basic design rules for implementing pn-junction photodiodes in deep submicron technologies have been outlined.

Finally, a biologically-inspired scheme to obtain optical information from vision chips has been presented. The technique uses a ultra-low power (20pJ per spike) spiking photoreceptor to output intensity information from a set of photodiodes. The scheme uses spike-interval coding to encode the information asynchronously and therefore aims to reduce coupled switching noise when distributed throughout a system.

References

[1] H.-S. Wong, "Technology and device scaling consideration for CMOS imagers," *IEEE Transactions on Electron Devices*, vol. 43, no. 12, pp. 2131–2142, 1996.

[2] T. Lulé, S. Benthien, H. Keller, F. Mütze, P. Rieve, K. Siebel, M. Sommer and M. Böhm, "Sensitivity of CMOS based imagers and scaling perspectives," *IEEE Transactions on Electron Devices*, vol. 47, no. 11, pp. 2110–2122, 2000.

[3] S. M. Sze, *Physics of Semiconductor Devices*. Wiley, 1981.

[4] J. Geist and H. Baltes, "High accuracy modeling of photodiode quantum efficiency," *Applied Optics*, vol. 28, no. 18, pp. 3929–3939, 1989.

[5] C. Kittel, *Introduction to Solid State Physics*. Wiley, 1995.

[6] W. B. Leigh, *Devices for Optoelectronics (Optical Engineering)*. Marcel Dekker, 1996.

[7] J. Singh, *Electronic and Optoelectronic Properties of Semiconductor Structures*. Cambridge University Press, 2003.

[8] J. S. Lee, R. I. Hornsey and D. Renshaw, "Analysis of CMOS Photodiodes. I. Quantum efficiency," *IEEE Transactions on Electron Devices*, vol. 50, no. 5, pp. 1233–1238, 2003.

[9] J. S. Lee, R. I. Hornsey and D. Renshaw, "Analysis of CMOS Photodiodes. II. Lateral photoresponse," *IEEE Transactions on Electron Devices*, vol. 50, no. 5, pp. 1239–1245, 2003.

[10] O. Yadid-Pecht and R. Etienne-Cummings, eds., *CMOS Imagers: From Phototransduction to Image Processing*. Kluwer Academic Publishers, 2004.

[11] G. I. T. D. S-C. Liu, J. Kramer and R. Douglas, *Analog VLSI: Circuits and Principles*. The MIT Press, 2002.

[12] A. Haapalinna, P. Karha and E. Ikonen, "Spectral Reflectance of Silicon Photodiodes," *Applied Optics*, vol. 37, no. 4, pp. 729–732, 1998.

[13] L. Polerecky, "Theoretical background for the measurements of refractive index and thickness of a thin dielectric layer," *Internal Dublin City University Report*, 1999.

[14] United Microelectronic Corporation (UMC), *0.18um Mixed Mode/RFCMOS Technology 1.8V/3.3V 1P6M Electrical Design Rule (with Metal/Metal Capacitor Module)*, 1.2p2 ed., 2003.

[15] M. Mahowald, *VLSI Analogs of Neuronal Visual Processing: A Sythesis of Form and Function*. PhD thesis, California Institute of Technology, Pasadena, California, 1992.

[16] T. Delbrck and C. A. Mead, "Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits," *Vision Chips: Implementing vision algorithms with analog VLSI circuits, by C. Koch and H. Li eds.*, pp. 139–161, 1995.

[17] S. Mendis, S. E. Kemeny and E. R. Fossum, "CMOS Active Pixel Image Sensor," *IEEE Transactions on Electron Devices*, vol. 41, no. 3, pp. 452–453, 1994.

[18] S. Decker, D. McGrath, K. Brehmer and C. G. Sodini, "A 256x256 CMOS imaging array with wide dynamic range pixels and column-parallel digital output," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 12, pp. 2081–2091, 1998.

[19] O. Yadid-Pecht and A. Belenky, "Autoscaling CMOS APS with customized increase of dynamic range," *IEEE International Solid-State Circuits Conference*, pp. 100–101, 2001.

[20] W. Yang, "Image sensor array with threshold voltage detectors and charged storage capacitors." US Patent Number 5,214,274, 1993.

[21] W. Yang, "A Wide-Dynamic-Range, Low-Power Photosensor Array," *Proceedings of IEEE International Solid-state Circuits Conference*, pp. 230–231, 1994.

[22] L. Qiang and J. G. Harris, "A time-based CMOS image sensor," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 840–843, 2004.

[23] J. Kramer, "An on/off transient imager with event-driven, asynchronous read-out," *IEEE International Symposium on Circuits and Systems*, vol. 2, p. 165168, 2002.

[24] T. G. Constandinou, P. Degenaar, D. Bradley and C. Toumazou, "An on/off spiking photoreceptor for adaptive ultrafast/ultrawide dynamic range vision chips," *Proceedings of the IEEE Workshop on Biomedical Circuits and Systems*, vol. S1, pp. 6–9, 2004.

[25] P. Abshire and A. Andreou, "Capacity and energy cost of information in biological and silicon photoreceptors," *Proceedings of IEEE Systems*, vol. 89, no. 7, pp. 1052–1064, 2001.

[26] C. A. Mead and M. Ismail (eds.), *Analog VLSI implementation of neural systems*. Kluwer Academic Publishers, 1989.

[27] K. A. Boahen, "A Retinomorphic Chip with Parallel Pathways: Encoding ON, OFF, INCREASING, and DECREASING Visual Signals," *Kluwer Analog Integrated Circuits and Signal Processing*, vol. 30, no. 2, pp. 121–135, 2002.

[28] R. G. Smith, N. K. Dhingra, Y. H. Kao, and P. Sterling, "How efficiently a ganglion cell codes the visual signal," *Proceedings of the 23rd Annual EMBS International Conference*, pp. 663–665, 2001.

[29] T. Delbruck and S. Liu, "A silicon early visual system as a model animal," *Vision Research*, vol. 44, no. 17, pp. 2083–2089, 2004.

[30] P. Lichtsteiner, T. Delbrck and J. Kramer, "Improved ON/OFF temporally differentiating address-event imager," *Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 211–214, 2004.

[31] A. Moini, ed., *Vision Chips*. Kluwer Academic Publishers, 1999.

[32] T. Y. W. Choi, B. E. Shi and K. A. Boahen, "An ON-OFF Orientation Selective Address Event Representation Image Transceiver Chip," *IEEE Transactions on Circuits and Systems I: Regular papers*, vol. 51, no. 2, pp. 342–353, 2004.

[33] D. K. Warland, P. A. Reinagel and M. Meister, "Decoding Visual Information from a population of retinal ganglion cells," *Journal of Neurophysiology*, vol. 78, no. 5, pp. 2336–2350, 1997.

[34] R. Wodnicki, G. W. Roberts and M. D. Levine, "A foveated image sensor in standard CMOS technology," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 357–360, 1995.

[35] R. Wodnicki, G. W. Roberts and M. D. Levine, "A log-polar image sensor fabricated in a standard 1.2um ASIC CMOS process," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1274–1277, 1997.

[36] G. Sandini, P. Questa, D. Scheffer, B. Diericks and A. Mannucci, "A retina-like CMOS sensor and its applications," *Proceedings of IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 514–519, 2000.

[37] G. Sandini, J. Santos-Victor, T. Paidia and F. Berton, "OMNIVIEWS: direct omnidirectional imaging based on a retina-like sensor," *Proceedings of IEEE Sensors*, vol. 1, pp. 27–30, 2002.

[38] A. Bernardino, J. Santos-Victor and G. Sandini, "Foveated active tracking with redundant 2D motion parameters," *Robotics and Autonomous Systems*, vol. 3, no. 4, pp. 205–221, 2002.

[39] R. Etienne-Cummings, J. Van der Spiegel, P. Mueller and Z. Mao-Zhu, "A foveated silicon retina for two-dimensional tracking," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 6, pp. 504–517, 2000.

[40] M. Azadmehr, J. P. Abrahamsen and P. Hafliger, "A Foveated AER Imager Chip," *IEEE International Symposium on Circuits and Systems*, pp. 2751–2754, 2005.

[41] A. Bermak, "A CMOS imager with PFM/PWM based analog-to-digital converter," *IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 53–56, 2002.

[42] A. Kitchen, A. Bermak and A. Bouzerdoum, "PWM digital pixel sensor based on asynchronous self-resetting scheme," *IEEE Electron Device Letters*, vol. 25, no. 7, pp. 471–473, 2004.

[43] E. Culurciello and R. Etienne-Cummings, "Second generation of high dynamic range, arbitrated digital imager," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 281–294, 2003.

[44] E. Culurciello and R. Etienne-Cummings, "Second generation of high dynamic range, arbitrated digital imager," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 828–831, 2004.

[45] W. P. Lee, C. T. Hsu, C. Y. Tsoi, A. Bermak and K. N. Leung, "Synchronization analysis in spiking pixel architecture-hardware implementation and mismatch analysis," *IEEE Region 10 Conference*, vol. D, pp. 274–277, 2004.

[46] Q. Xin, G. Xiaochuan and J. G. Harris, "A time-to-first spike CMOS imager," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 824–827, 2004.

[47] A. Bermak and C. Shoushun, "A Low Power CMOS Imager based on Time-to-First-Spike encoding and Fair AER," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 5306–5309, 2005.

Chapter 6

ORASIS: A Micropower Centroiding Vision Processor

6.1 Introduction

This chapter presents a computationally-efficient vision processing chip for multi-object-based centroiding and sizing. The outlined system, named ORASIS; constitutes a 48×48 pixel photo-detecting and distributed processing array. This directly implements the bio-pulsating contour reduction algorithm (Chapter 4) using hybrid cellular topologies involving weak-inversion analogue and asynchronous digital circuit techniques (Chapter 2).

The presented system provides the following additional functionality to previous work developed in this area (reviewed in Chapter 3).

1. Object centroiding: This system is the *first* developed (to date) capable of facilitating simultaneous centroid detection of unlimited[1] objects.
2. Object Sizing: This system is the *first* developed (to date) capable of facilitating parallel size determination of unlimited[1] objects.
3. Object counting: This system is the *first* developed (to date) capable of parallel object counting.
4. Input Versatility: The system can be configured such that it can operate with a wide

[1]There is no maximum object object constraint, i.e. concerning the amount of objects that can be processed in parallel. The only limiting factor is the address event communication capacity.

variety of different input image types. The image processing parameters that can be tuned are: edge detection threshold, object/background threshold sense and threshold offset.

5. Ultra-low power consumption: The developed system achieves relatively high computational efficiency in comparison to traditional techniques.

This chapter begins with a top-level system architecture, describing how the various blocks are hierarchically arranged, interconnected and can be scaled. Following is the cellular (tessellating) organisation outlining the functional sub-blocks for implementing the given algorithm. These sub-blocks are then each described in detail, including circuit schematics with accompanying results. The fabricated prototypes are then discussed, with a brief overview of their structure and contents. Finally system-level results, both simulated and measured conclude the system description.

6.2 System Organisation

Based on implementing the bio-pulsating contour reduction algorithm; described previously in Chapter 4, a system architecture is outlined [1], shown in Fig. 6.1.

6.2.1 Pixel Array

This can be subdivided into four "corners", each of $(x/2, y/2)$ dimensions, where (x,y) is the array size (48x48). These sub-blocks are interconnected in the same way pixel elements are interconnected internally with the exception of power supplies, bias currents and control signals. Furthermore each "corner" is synthesised using a unique set of pixel types, depending on location. For example, the top-left corner block will contain *left-edge*, *top-left-corner*, *top-edge* and *standard* pixel types. In total there are nine different pixel types, i.e. four edge pixels, four corner pixels and one standard (centre) pixel. The difference between these pixel types is simply due to the different terminating edge configurations, for example a corner cell will have two terminating edges, whereas an edge cell will only have one terminating edge. As will be seen later, the I/O connections need to be terminated correctly for useful and error-free operation.

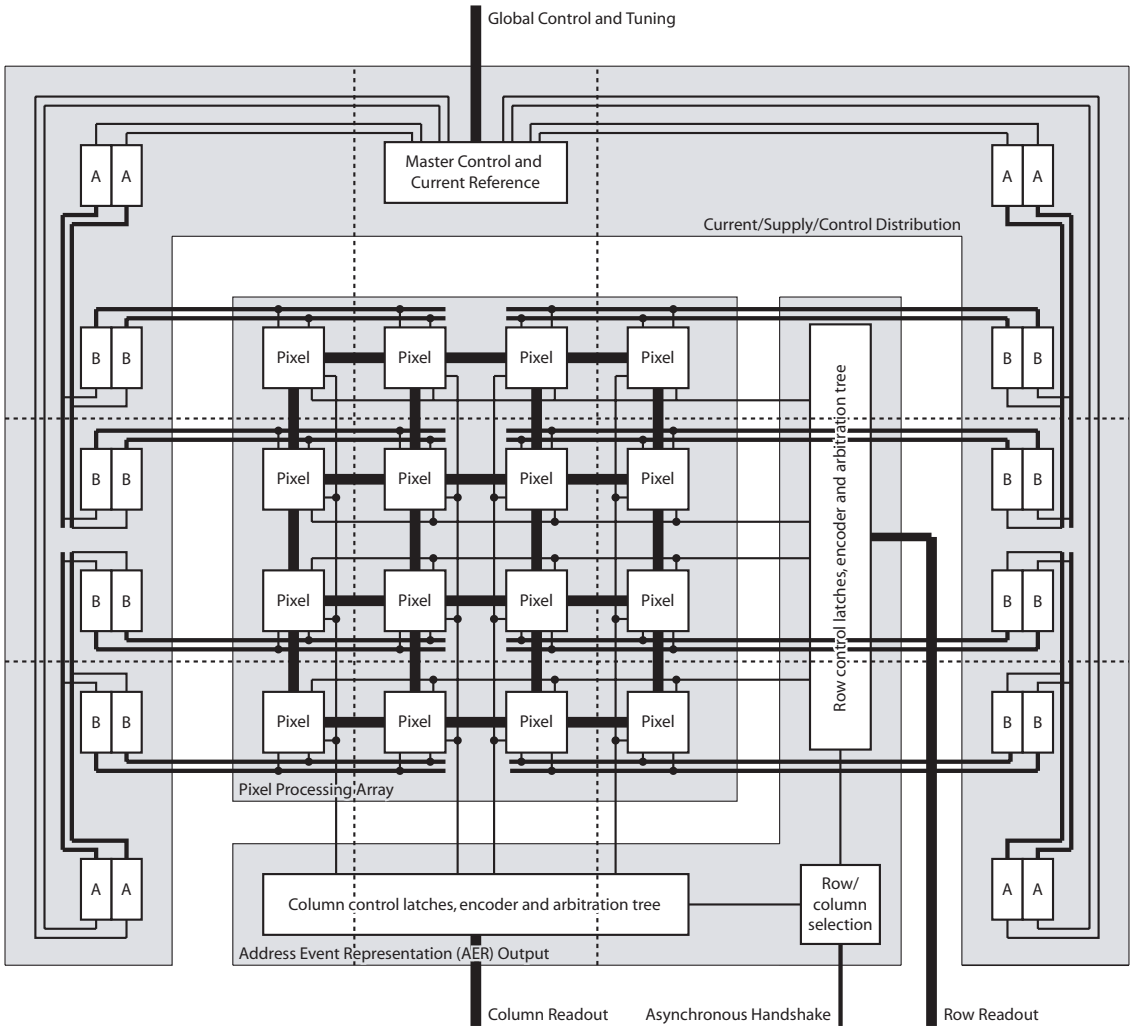


Figure 6.1: The proposed ORASIS System architecture. Illustrated are the three main components: pixel processing array, address event representation readout and current/supply/control distribution tree. The dotted lines represent the "stretch" marks, i.e. how the system can be scaled to a larger size array.

6.2.2 Global Signal Distribution

Due to the large number of in-pixel *processors*, a current distribution scheme is needed for bias current copying and hierarchical fanout required for distribution of digital (control) signals.

For the given size of pixel array (48x48), the following four-level distribution tree/fanout is proposed for current distribution:

- Corner (1 to 4): Four initial master bias currents are generated and used to supply the four corner headers of the pixel array, i.e. one header in each corner block (shown in Fig. 6.1: sub-block A).
- Row (1 to 24): Each master reference set is then used to make (y/2) copies feeding every row header in its corner (shown in Fig. 6.1: sub-block B).
- Column (1 to 24): Each row header in turn is used to make (x/2) copies feeding every pixel in its row.
- Pixel (1 to 4): Within each pixel these bias currents are locally combined and copied further (discussed later).

Current-mode vs. Voltage-mode Current Distribution

In a current-mode scheme, at each current-distribution chain, the currents are copied locally using devices in close proximity (thus well-matched) and subsequently distributed along separate metal lines. This increases device count and metal area usage in comparison to a voltage-mode current distribution scheme.

A voltage-mode scheme uses voltage distribution to set device input voltages (over a large area) and thus creates the bias currents. However, the error contribution is two-fold; systematic in addition to increased mismatch. Using set bias voltage distribution, metal line resistance over a relatively large distance results in a linear voltage gradient. When used to generate bias currents, this translates in a non-linear current gradient through device transconductance. Furthermore the mismatch increases due to large proximity device separation where process variation gradients come into effect.

Therefore current-mode current distribution is preferred over voltage-mode distribution for large area distribution to improve current matching at the expense of silicon area.

Digital Fanout (Control Distribution)

The scheme used for the control signal fanout is arranged using the same tree hierarchy as for the current distribution. The digital signals are buffered at each level, by means of a quad inverter cascade of increasing dimensions. By designing the buffers output transistors to be relatively large, they can drive many relatively small input transistors in subsequent levels, therefore ensuring fast and reliable operation.

System Scalability

This proposed architecture allows for a certain degree of scalability (perhaps tenfold), however to scale to much larger array sizes, for example, a 1 megapixel array, the distribution hierarchy would have to be modified, i.e. increased number of levels/duplication stages. The pixel circuitry and address-event hardware are however fully scalable.

6.2.3 System Input/Output

The various I/O signals used to control and tune the system and subsequently communicate processed data off-chip are defined below:

- GLOBAL_RESET: For initialisation of the initial state of the pixel array. Resets all the distributed memory contents.
- LOCAL_RESET: Defines whether localised resetting is enabled, realising a pulsating action, or operating in single-shot mode; initiated through a global reset signal.
- THRES_MODE: Defines whether background intensity is above (or below) object intensity.
- OUTPUT_SEL[1:0]: Selects signal to be routed to AER output: Centre, State and Reset.
- GLOBAL_AVERAGE: Selects whether the wide-field local average is computed as a column average or global average.

- IGLOBAL: Adjusts the global average level by sinking or sourcing a correction current to this node when GLOBAL_AVERAGE is asserted.
- IBIAS: Provides edge detector bias current and defines artificial propagation delay constant.
- ITUNE: Tunes edge detector sensitivity, i.e. threshold of flagging edge detection.
- CHIP_REQ: Chip request for off-chip communication; used to signal asynchronous handshake to receiving device on having data ready for transmission.
- CHIP_ACK: Chip acknowledge for off-chip communication; used to acknowledge successful transmission of data from receiving device.
- X[5:0]: The X-coordinate of address being transmitted.
- Y[5:0]: The Y-coordinate of address being transmitted.

6.2.4 Pixel Organisation

The basic pixel organisation is illustrated in Fig. 6.2. Based on the bio-pulsating contour reduction algorithm described in Chapter 4, this architecture realises a direct implementation.

The photodiode is a reverse biased n-well/p-substrate junction (discussed in Chapter 5), of dimensions $30\mu m \times 30\mu m$. This feeds the in-pixel analogue signal processing (ASP) core which smooths, averages, compares and thresholds as described earlier. The ASP generates two signals; CONTOUR and THRESHOLD, which in turn are used to feed the asynchronous binary processing (ABP) core. This facilitates the contour reduction through asynchronous signal propagation, flagging the centres on detection. On centroid detection, the output neuron negotiates with the Address Event Representation (AER) core for a timing slot for off-chip communication.

The following three sections describe the structure and make-up of these blocks (ASP, ABP and AER), the fundamental circuit theory, circuit operation and simulated / measured results.

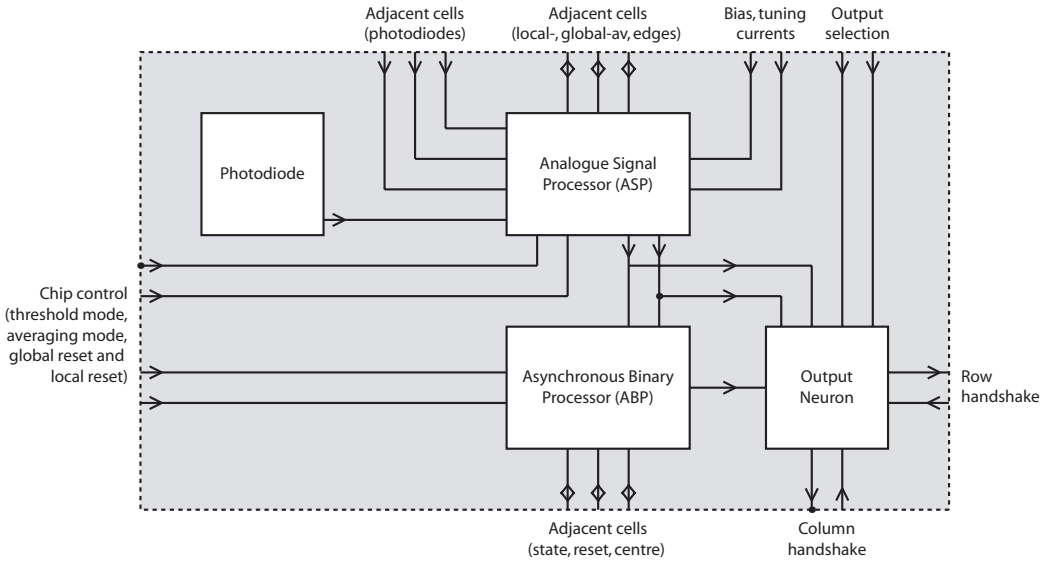


Figure 6.2: The proposed ORASIS Pixel architecture. Illustrated are the four main components: sensor (photodiode), Analogue Signal Processor (ASP), Asynchronous Binary Processor (ABP) and Address Event Representation (AER) neuron.

6.3 Distributed Analogue Signal Processing (ASP) Core

The concept of using a distributed ASP core to feature extract is to avoid use of analogue-to-digital converters (ADC) to reduce power consumption (previously discussed more rigorously in Chapter 2). Instead analogue processing is used to reduce the computation to a series of comparisons, where simple comparators, i.e. 1-bit converters can be used for discrete, asynchronous output.

This section describes the distributed architecture and circuits for extracting the required CONTOUR and THRESHOLD signals from a matrix of photocurrents.

6.3.1 Architecture

External

The extracellular interconnectivity is illustrated in Fig. 6.3 and previously in Fig. 4.5. Each pixel-cell requires 16 connections with adjacent cells to achieve the required computational functionality (front-end image processing):

- Narrow-field local averaging (6 connections): Each cell receives three *iphoto_in* current inputs from adjacent photodiodes (lower, right and lower-right) and consequently transmits its *iphoto_out* current to three adjacent cells (left, upper and upper-left).
- Wide-field local averaging (2 connections): as column averaging is used, every cell connects to a column averaging node shared with the upper and lower cells.
- Edge detection (4 connections): Every pixel-cell receives two *vphoto_in* inputs from adjacent photodiodes (lower and right) and consequently transmits its *vphoto_out* value to two adjacent cells (left and upper).
- Contour detection (4 connections): As each cell compares its photo-intensity with that of the cell adjacently below and to the right, two edges are computed per pixel. Therefore to process all edges adjacent to a photodiode two edge inputs (from left and upper cells) and two edge outputs (to right and lower cells) are required.

Internal

The internal architecture of the ASP core is illustrated in Fig. 6.4.

The ASP core consists of three main (functional) blocks for: (1) averaging and comparison, (2) edge detection and (3) contour discrimination, in addition to some support circuitry.

A cells photo-voltage (V_{photo}); being a log-compression of its photocurrent, is generated at the front-end in the averaging and comparison block. This block computes the narrow-field and wide-field local averages, compares these and thresholds to determine whether a certain pixel is above or below the average intensity level ($V_{threshold}$). In parallel, the cells photo-voltage is compared with those of neighbouring cells to determine whether it lies on an edge. The four edge signals (to adjacent photodiodes) are then used in conjunction with the THRESHOLD signal to determine whether a cell satisfies the CONTOUR condition.

6.3.2 Averaging and Comparison

The schematic diagram for the ASP block responsible for photodetection, averaging, comparison is given in Fig. 6.5.

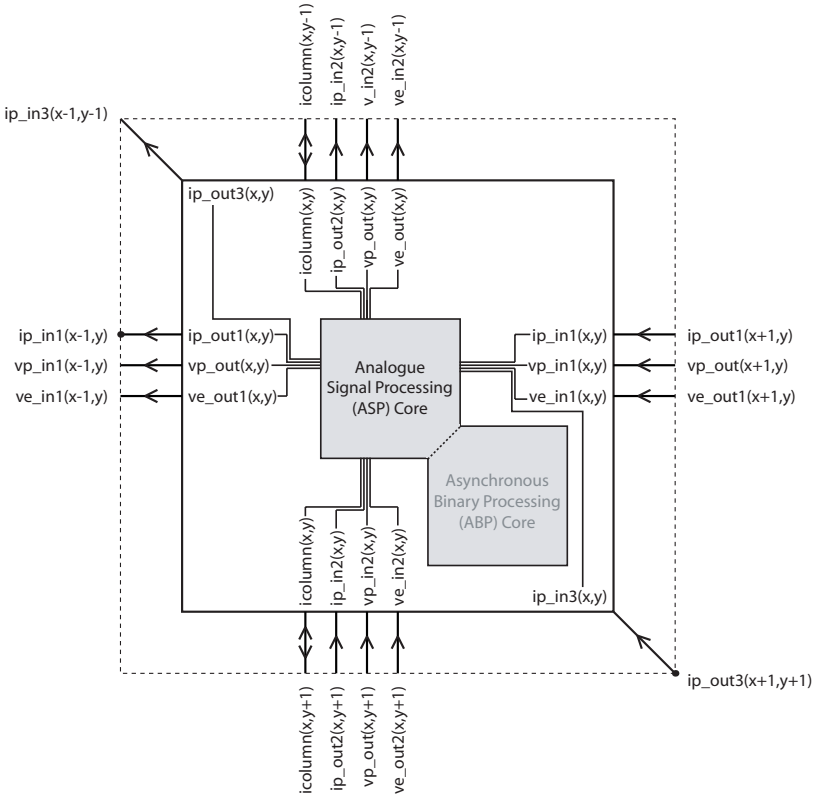


Figure 6.3: Symbol representation of the analogue signal processing (ASP) core. Illustrated is the external connectivity of analogue signals, i.e. with other cells, showing the requirements for cellular tessellation. This excludes global control signals and bias point connections. Nodes have been abbreviated for clarity as follows: vp=vphoto, ip=iphoto, ve=vedge.

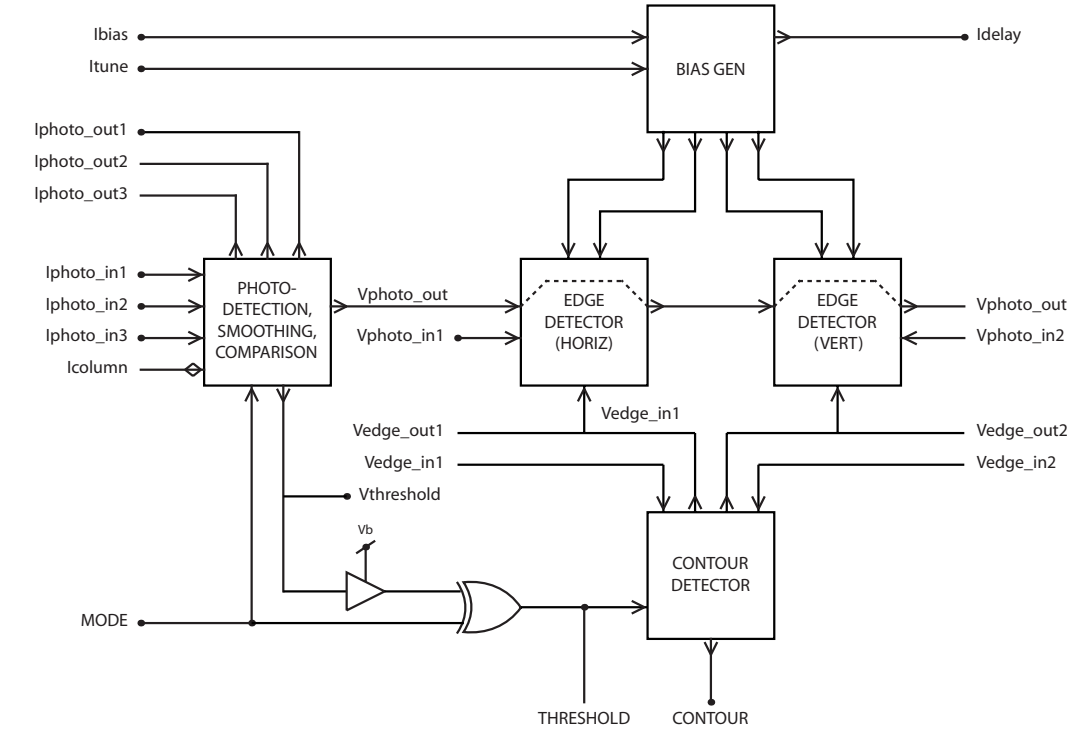


Figure 6.4: Schematic diagram of the in-pixel analogue signal processing (ASP) organisation. Illustrated is the internal connectivity emphasising the signal flow path between the various blocks.

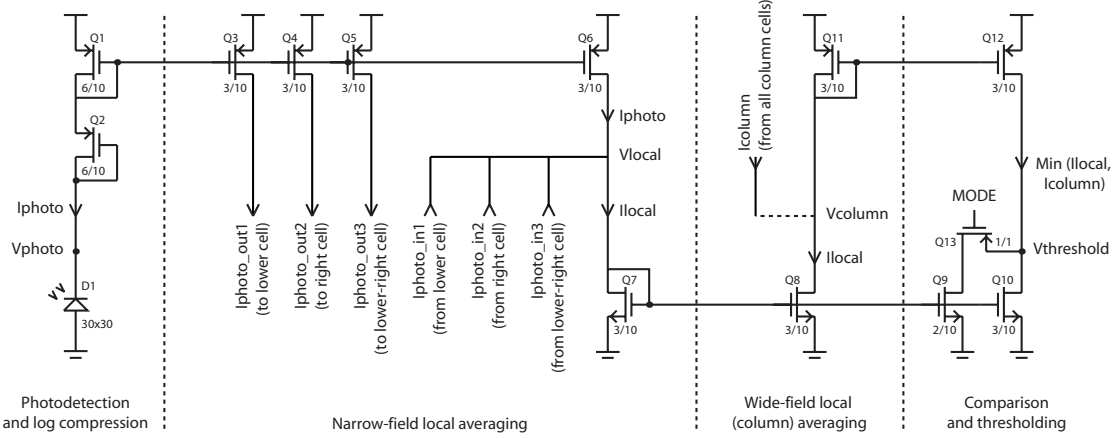


Figure 6.5: Schematic diagram of the photodiode interface circuit including current-mode narrow- and wide-field averaging/smoothing and comparison.

Phototransduction: The pn-junction photodiode (D1) is reverse-biased by stacking two diode-connected PMOS devices (Q1, Q2) to V_{dd}. The photocurrent range for the given device under the expected light levels is from 100fA (dark current) to 5nA. For this current range devices Q1 and Q2 operate in the weak inversion region therefore the applied reverse-bias is logarithmically proportional to the photocurrent [2, 3] as expressed in Eqn. 6.1.

$$V_{photo} = V_{dd} - (V_{gs1} + V_{gs2}) \approx V_{dd} - 2n\phi_t ln\left(\frac{I_{photo}}{I_0}\right) \tag{6.1}$$

Where: n is the subthreshold slope factor, ϕ_t is the thermal voltage, I_{photo} is the photocurrent and I_0 is the pre-exponential current.

Narrow-field local averaging: The node in-between the stacked devices is also used to form a current mirror with devices Q3-Q6; providing scaled, copied currents for current-mode averaging. Devices Q3-Q5 source copied photocurrents ($I_{photo\_out1,2,3}$) to adjacent cells and device Q6 receives and sums copied photocurrents from adjacent cells to form a four-pixel average current, such that $I_{local} = (I_{photo} + I_{photo\_out1} + I_{photo\_out2} + I_{photo\_out3})/2$.

Wide-field local averaging: The wide-field local average is implemented by using a column averaging technique. This is facilitated by summing all the copied (through current mirror Q7, Q8) narrow-field smoothed currents (I_{local}). Normalisation is then achievable by copying this current using a distributed 1:1 current mirror per cell. This has the effect of forming an X:1 scaled mirror; with X being the number of cells attached to the column (see Fig. 6.6).

Current-mode comparison: The near-field local (cellular) average is then compared to the wide-field local (column) average by means of a basic current comparator formed by an opposing source/sink transistor pair [4] (Fig. 6.5: Q10, Q12). If the device bias points are similar, then both devices will be in saturation and the output voltage ($V_{threshold}$) is given in Eqn. 6.3.

$$V_{threshold} = \frac{I_{column}\,(1 + \lambda_p V_{dd})}{I_{local}\,(1 + \lambda_n) + I_{column}\lambda_p} \tag{6.2}$$

Where: λ_n and λ_p are the linear channel length modulation (early) factors for N- and P-MOS devices respectively. Subsequently, if the input currents are exactly equal, then this simplifies to:

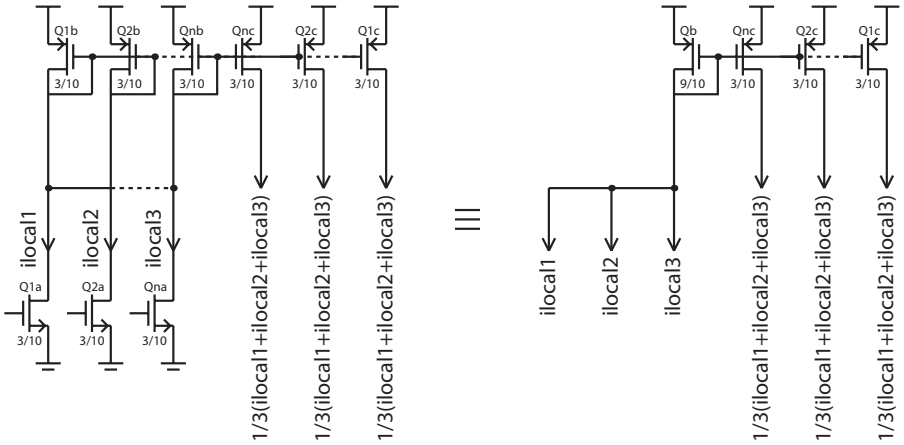


Figure 6.6: Schematic representation of the wide-field (column) averaging mechanism. Example given for a three row example.

$$V_{threshold} = \frac{1 + \lambda_p V_{dd}}{1 + (\lambda_n + \lambda_p)} \tag{6.3}$$

However, if the source/sink bias points (saturation currents) are different, the device with the higher bias point will be forced out of saturation, i.e. it will enter the ohmic region. For example, in Fig. 6.5, if $I_{column} > Ilocal$, the source device (Q12) will operate in the linear region, and the output voltage ($V_{threshold}$) will swing upwards towards Vdd. This behaviour is described by eqn. 6.4 (for $I_{local} < I_{column}$) and eqn. 6.5 (for $I_{local} > I_{column}$).

$$V_{threshold} = \frac{I_{column}\,(1 + \lambda_p V_{dd})}{I_{column}\lambda_p + \frac{I_{local}}{\phi_t}} \tag{6.4}$$

$$V_{threshold} = \frac{I_{local}V_{dd} - I_{column}\phi_t}{I_{local} + I_{column}\lambda_n\phi_t} \tag{6.5}$$

Although the generated threshold voltage ($V_{threshold}$) describes the current comparison discretely for a substantial differential current, smaller current differences cause $V_{threshold}$ to remain between Vss and Vdd. For driving CMOS static logic, such a signal is undesirable, for a non-discrete input can give rise to large "short-circuit" currents. Therefore a thresholding buffer is required to "square up" this signal to reliable discrete levels (discussed later).

Threshold mode option: To provide versatility to a wider range of image types, a

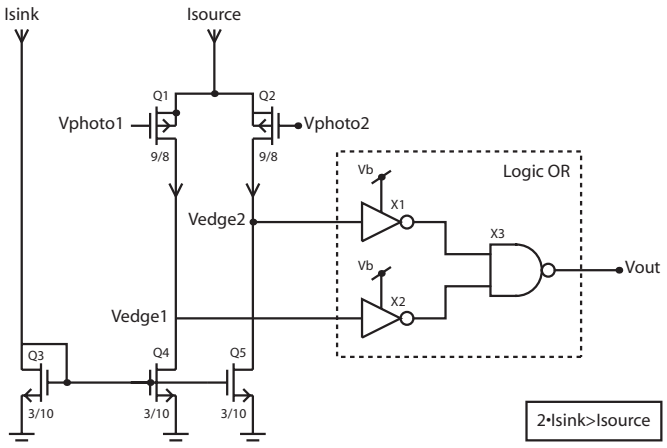


Figure 6.7: Schematic diagram of the tunable discrete edge detector circuit. Details of the current generation scheme (I_{source} and I_{sink}) and implementation of thresholding inverters (X1 and X2) are provided later.

threshold mode option can provide the functionality to select between light object on dark background and dark object on light background. To achieve this, the threshold detection offset must be shifted (inverted) to provide the correct margin for adequate noise rejection and robustness to process variation. This can be implemented by means of altering the effective device aspect ratios by switching in additional devices in parallel with the sinking device (Fig. 6.5: Q9, Q13). Furthermore the discrete output requires to be inverted, easily achievable by using a XOR gate (see Fig. 6.4).

Global averaging option: Using this current-mode averaging/thresholding scheme, it is easily extendable to provide the option for global averaging, with an input for threshold correction. This is achievable by switching all the column averaging nodes (V_{column}) to be shorted together thus realising a single global average. Subsequently, if an external current is sourced or sunk to this node, this will have the effect of adjusting the global average threshold higher or lower.

6.3.3 Edge Detecting and Contour Discrimination

The schematic given in Fig. 6.7 illustrates the edge detection circuitry [5] [6] inserted between *every* pair of adjacent pixels.

Edge Detection: Principle of Operation

The diode-connected devices (previously illustrated in Fig. 6.5) provide the logarithmically compressed voltage inputs (V_{photo1} and V_{photo2}). Two diode-connected devices are used to ensure the current sourcing device (not shown) remains in saturation for small photocurrents. The differential voltage (V_{photo1}, V_{photo2}) is applied to the PMOS differential pair (Q1 and Q2) sourced by the current I_{source}. The differential pair tail currents are sunk via the current mirror (Q3, Q4 and Q5) which is controlled by the sink current; I_{sink}. The operation is as follows:

- I_{source} is selected such that it generates sufficient transconductance and therefore gain to ensure reliable operation for the minimum response time required (limited by response of photodiodes). Too high a value would result in a useless increase in power consumption (for this is static power dissipation) with no increase in system performance. Furthermore, both the bias currents and device sizes are selected such that all the devices remain (as much as possible) within a single region of operation; for the expected current levels, in weak inversion. This is to avoid any asymmetric behaviour due to some devices having a significant drift-current influence, i.e. entering moderate inversion.

- I_{sink} is adjusted to lie in between $I_{source}/2$ and I_{source} and sets the allowed tolerance before indicating an edge and flagging it up. This will set the gate-source voltages of devices Q4 and Q5. This voltage will in turn determine the maximum current that can be sunk from the drains of Q4 and Q5 (I_{d4max} and I_{d5max} respectively). Assuming devices Q1 and Q2 are ideally matched, this circuit operates in one of two states:

  1. ($V_{photo1} = V_{photo2}$): Since $I_{source}/2 < I_{sink} < I_{source}$ then $I_{d1} < I_{d8max}$ causing device Q4 to be forced into the ohmic region. This in turn will cause V_{edge1} to sit barely above ground and similarly Q5, I_{d6} and V_{edge2} will behave in the same way. As a result of V_{edge1} and V_{edge2} both being low, V_{out} will output high indicating there is no edge.
  2. ($V_{photo1} \neq V_{photo2}$): For example, if $V_{photo1} < V_{photo2}$ such that $I_{d1} = I_{d4max}$ then device Q4 is in saturation and V_{edge1} rises to just below V_{dd}. However $I_{d2} < I_{d5max}$ so device Q5 is still in the ohmic region, keeping V_{edge2} low. This will result in V_{out} outputting low indicating there is an edge.

Edge Detection: Circuit Analysis

Assuming devices Q1 and Q2 are operating in saturation, the following expression (Eqn. 6.6) can be derived, expressing the output current (differential).

$$I_{d1} - I_{d2} = I_{source} \cdot tanh\left(\frac{V_{photo1} - V_{photo2}}{2n\phi_t}\right) \tag{6.6}$$

Where: n is the charge effect due to the substrate (also referred to as the slope factor or subthreshold constant) and ϕ_t is the thermal voltage ($\phi_t = kT/q$=25.9mV at room temperature).

This can be split as to provide the single-ended tail currents described in expressions 6.7 and 6.8.

$$I_{d1} = \frac{1}{2} \cdot I_{source} \cdot \left[1 + tanh\left(\frac{V_{photo1} - V_{photo2}}{n\phi_t}\right)\right] \tag{6.7}$$

$$I_{d2} = \frac{1}{2} \cdot I_{source} \cdot \left[1 - tanh\left(\frac{V_{photo1} - V_{photo2}}{n\phi_t}\right)\right] \tag{6.8}$$

From 6.6, the large signal (6.9) and small signal (6.10) transconductance of the differential pair can be derived [7].

$$G_M = \frac{d(I1 - I2)}{d(V1 - V2)} = \frac{I_{source}}{2n\phi_t} \cdot sec^2\left(\frac{V_{photo1} - V_{photo2}}{2n\phi_t}\right) \tag{6.9}$$

$$g_m = \frac{I_{source}}{2n\phi_t} \tag{6.10}$$

Furthermore, expression 6.9 can be used to express the range of values for which the circuit will flag an edge detected.

$$I_{sink} - [G_M(V_{error} + |V_{photo1} - V_{photo2}|)] < 0 \tag{6.11}$$

Where: V_{error} is the error term expressing the total mismatch error in the differential pair

as an input referred voltage. This expression directly links into those developed previously for algorithm-based edge detection robustness in Eqns. 4.16 and 4.17.

Although Eqn. 6.11 yields at least one discrete edge signal (V_{edge1}, V_{edge2}) for a moderate differential input voltage ($V_{photo1} - V_{photo2}$), smaller variations result in graded output levels. This is due to the differential pair having a finite gain and thus operating as a transconductor rather than comparator. As a result, devices Q1-Q5 are in saturation and the edge output voltages are given by Eqns. 6.12 and 6.13.

$$V_{edge1} = \frac{1}{\lambda_n}\frac{I_{sink}}{I_{source}}\left(1 + e^{\frac{V_{photo2}-V_{photo1}}{n\phi_t}} - \frac{I_{source}}{I_{sink}}\right) \tag{6.12}$$

$$V_{edge2} = \frac{1}{\lambda_n}\frac{I_{sink}}{I_{source}}\left(1 + e^{\frac{V_{photo1}-V_{photo2}}{n\phi_t}} - \frac{I_{source}}{I_{sink}}\right) \tag{6.13}$$

Where: λ_n is the linear channel length modulation (early) factor for the sinking devices, assuming $V_{ds} > 3\phi_t$ (for device Q3-Q5). This expression ignores any current source non-idealities (early effect).

As discussed earlier (in threshold detection), such intermediate (non-discrete) voltage levels are undesirable for driving CMOS logic. Therefore thresholding buffers are inserted in between the edge signals (V_{edge1} and V_{edge2}) and logic NOR gate.

Edge Detection: Results

The intended edge detection functionality is illustrated in the simulation results given in Figs. 6.8 and 6.9. Figure 6.8 shows the operation of the circuit for set bias currents (I_{source} and I_{sink}), but with varying photocurrents (I_{photo1} and I_{photo2}). Results are given for ten different I_{photo1} levels spanning over three orders of magnitude, with I_{photo2} swept over the entire range (X-axis). The "window" of edge-detection is seen (fig. 6.8(e)) to exhibit excellent linearity throughout the tested photo-intensity range. Figure 6.9 shows the operation of the circuit for a set bias current I_{source} and photocurrent I_{photo1} with varying bias current I_{sink}, with I_{photo2} swept over the entire range (X-axis). This demonstrates the tunability of the edge detection window using a single current-mode input.

Furthermore, the current consumption profile for a 1nA bias, is illustrated in the simulated results (Figs. 6.8(f) and 6.9(f)). This shows an average 3.5nA total current consump-

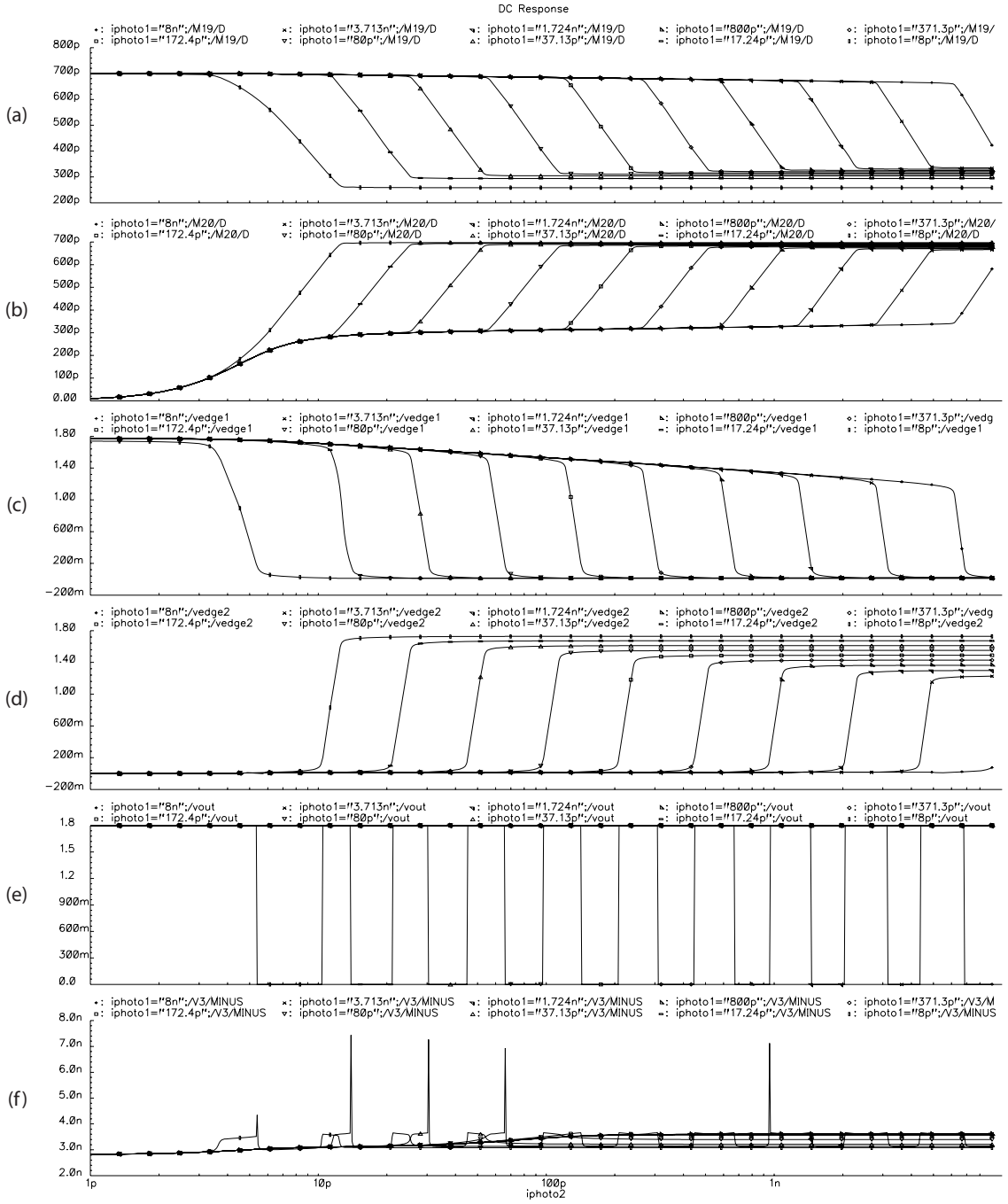


Figure 6.8: Simulation results of the edge detector circuit illustrating the discrete detection at varying light intensities. Results are for: $I_{source} = 1nA$, $I_{sink} = 600pA$, with $1pA \leq I_{photo1}, I_{photo2} \leq 10nA$. Shown (from top to bottom) are: (a) I_{d1}, (b) I_{d2}, (c) V_{edge1}, (d) V_{edge2}, (e) V_{out} and (f) I_{vdd}

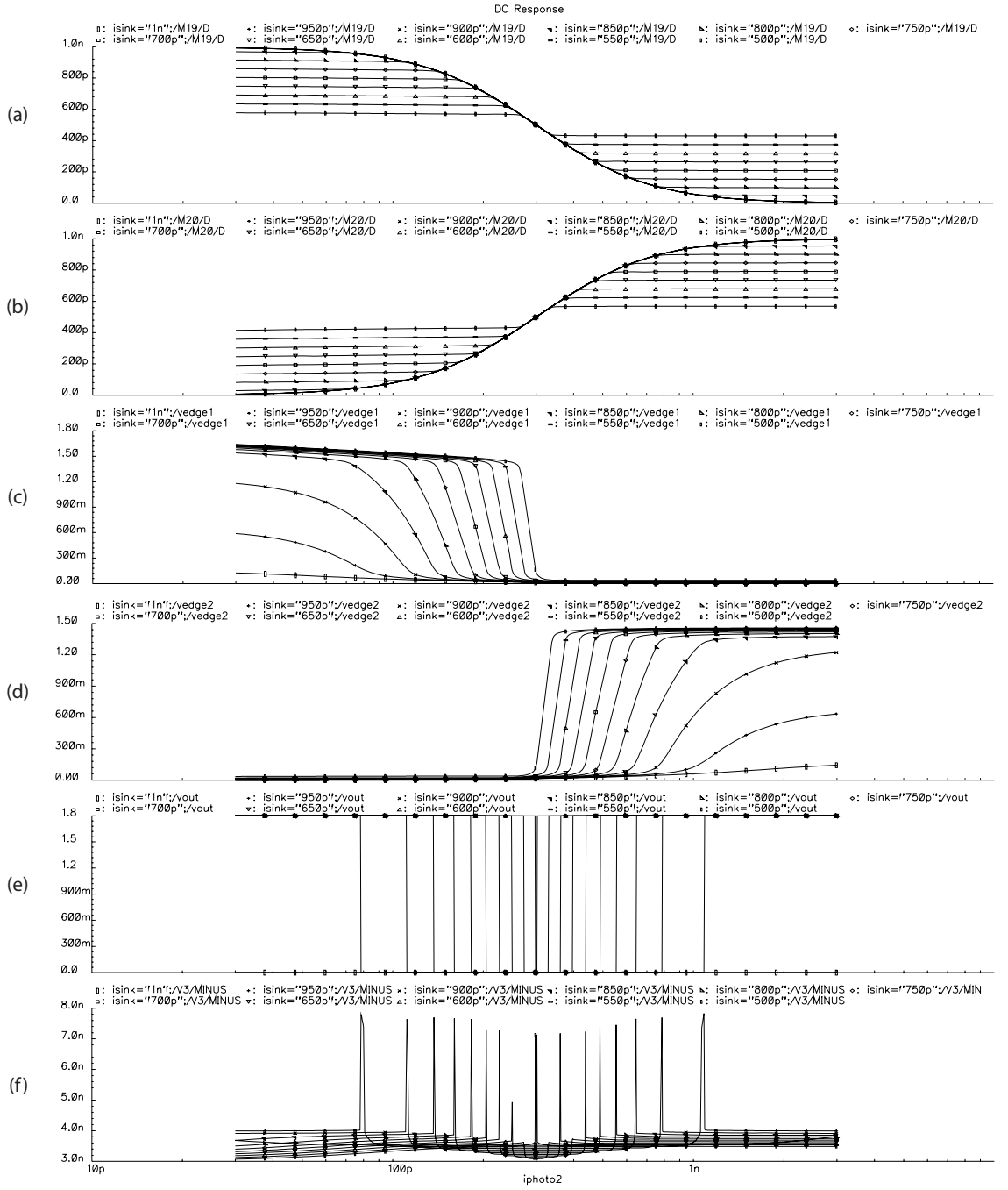


Figure 6.9: Simulation results of the edge detector circuit illustrating the tunable sensitivity. Results are for: $I_{source} = 1nA$, $500pA \leq I_{sink} \leq 1nA$, $I_{photo1} = 300pA$, with $1pA \leq I_{photo2} \leq 10nA$. Shown (from top to bottom) are: (a) I_{d1}, (b) I_{d2}, (c) V_{edge1}, (d) V_{edge2}, (e) V_{out} and (f) I_{vdd}

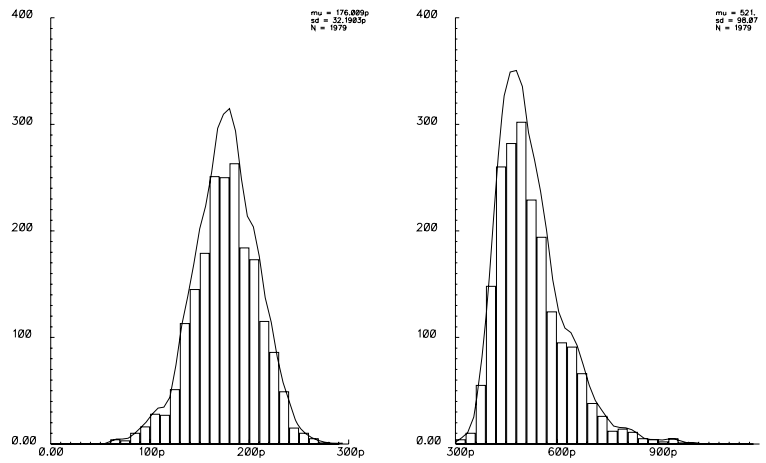


Figure 6.10: Monte Carlo simulation results for the edge detector illustrating variability of edge detection window to process variation and mismatch. For $I_{source} = 2nA$, $I_{sink} = 1.5nA$, $I_{photo1} = 300pA$, $1pA \leq I_{photo2} \leq 10nA$, statistical simulation of $N = 1979$ runs results in: $\mu_{lower} = 176.009pA$, $\sigma_{lower} = 32.190pA$, $\mu_{upper} = 521.234pA$, $\sigma_{upper} = 98.071pA$

tion (excluding photocurrents), over all operating regions with a peak consumption of 7-8nA at the onset of edge detection. Therefore the total average power consumption, per edge detector block is 6.5nW. The peaking mentioned previously is due to the current-limiting operation in the thresholding inverters for intermediate input voltages (discussed later).

Statistical simulation results show acceptable variability in edge detection threshold with process variation and device mismatch. The histogram given in Fig. 6.10 shows the full range ($\pm 3\sigma$) variation of current level to be 75-275pA for lower edge threshold and 325-850pA for upper edge threshold, for the given example. Although for the chosen device sizing, the statistical spread is not optimal (i.e. the design criteria are for power and area optimisation), the observed non-overlap band between upper and lower edge threshold levels indicates good yield and robust operation.

Contour Discrimination

The schematic given in Fig. 6.11 illustrates the contour discrimination logic present in every cell. This takes its (edge) inputs from the two internal (to that cell) edge detectors (right and bottom edges), the cell to the left (left edge) and the cell above (upper edge). Furthermore, the threshold input effectively inhibits static (spatial) noise from affecting the signal propagation within objects (discussed previously in Chapter 4).

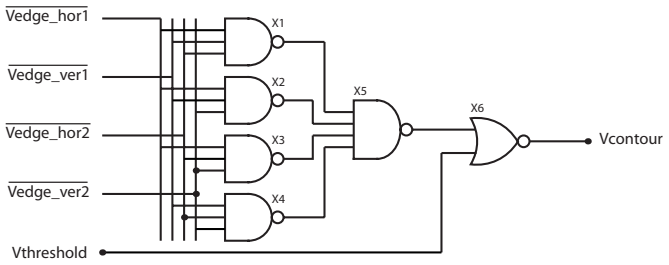


Figure 6.11: Schematic diagram of the contour discrimination combinational logic.

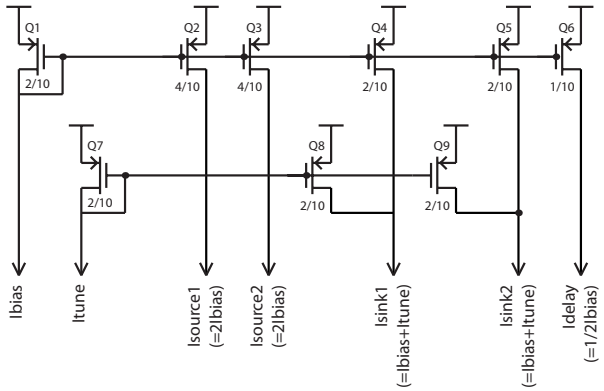


Figure 6.12: Schematic diagram of the in-pixel current distribution circuit, providing bias currents to the edge detecting and timing delay blocks.

6.3.4 Bias Distribution

Each pixel receives two individual bias currents; I_{bias} and I_{tune}. These are used to generate all accurate[2] in-pixel bias currents through current summation of copied currents (see Fig. 6.12). This generates $I_{source1} = I_{source2} = (2I_{bias})$, $I_{sink1} = I_{sink2} = (I_{bias} + I_{tune})$ and $I_{delay} = (I_{bias}/2)$.

6.3.5 Thresholding

To achieve power-efficient binary extraction functionality as previously mentioned, a thresholding inverter has been strategically inserted in the edge detection and local/global- averaging/comparison functions. This has the task of converting an analogue voltage to discrete level (1-bit conversion) with minimum power consumption. The implemented circuit solu-

[2]refers to relatively accurate currents (± 5%), in comparison to voltage-mode distributed currents (± 25%)

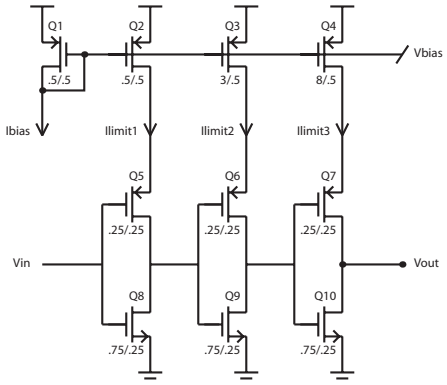


Figure 6.13: Schematic diagram of the current-limiting thresholding inverter; used for 1-bit conversion.

tion is illustrated in Fig. 6.13.

The basic concept is to threshold the signal using a three-stage cascade of logic inverters. Since the "short-circuit" voltage component would be prohibitively high in directly implementing this, the inverters power-supply (V_{dd}) is connected through a current-sourcing device. This has the effect of current limiting the short-circuit current at each stage. By scaling this limiting current, a successively *squarer* signal is obtained in successive stages. Furthermore, as the circuit topologies demand a low threshold detection point ($\approx V_{dd}/6$) a scaled PMOS current (limiting) mirror can help achieve this low threshold point. For a 1nA initial stage bias, the optimum (regarding power consumption) current-limit ratio is 1:6:16.

Statistical simulation results show acceptable variability in threshold voltage with process variation and device mismatch. The histogram given in Fig. 6.14 shows the full range ($\pm 3\sigma$) variation of threshold voltage to be 200-350mV. Although this represents a significant 8.3% variation over the power supply range, the target input signals (V_{edge1}, V_{edge2} and $V_{threshold}$) exhibit a steep roll-off in this region and therefore some variation in threshold level will translate to negligible output referred error.

6.4 Distributed Asynchronous Binary Processing (ABP) Core

Having extracted the important features by means of the ASP core, the image data has been reduced to two binary bits per pixel; CONTOUR and THRESHOLD. Subsequently, using these inputs to feed and synchronise a distributed, asynchronous binary network,

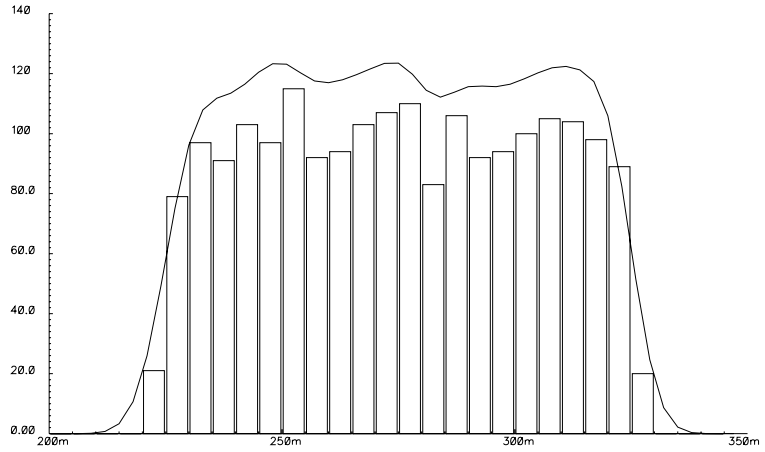


Figure 6.14: Monte Carlo simulation results for the thresholding inverter illustrating variability of threshold voltage to process variation and mismatch. $\mu = 275.322m$, $\sigma = 29.028m$, $N = 2000$

computationally-efficient spatiotemporal processing can be achieved on another level.

This section describes the distributed architecture and combinational circuits for facilitating the object segmentation and centroid extraction from a matrix of CENTROID and THRESHOLD binary inputs.

6.4.1 Architecture

External

The extracellular interconnectivity is illustrated in Fig. 6.15 and previously in 4.4. Each pixel-cell requires 52 connections with adjacent cells to achieve the required computational functionality (segmentation and centroiding):

- STATE (12 connections): Each cell receives eight STATE inputs from adjacent cells (four from directly adjacent and four from a three-cell proximity, i.e. three pixels to the right, three pixels above, etc) and consequently transmits its STATE contents to its four directly adjacent cells.
- RESET (8 connections): Each cell receives four RESET inputs from directly adjacent cells and transmits its RESET status back to them.
- CENTRE (16 connections): Each cell receives eight CENTRE inputs from adjacent

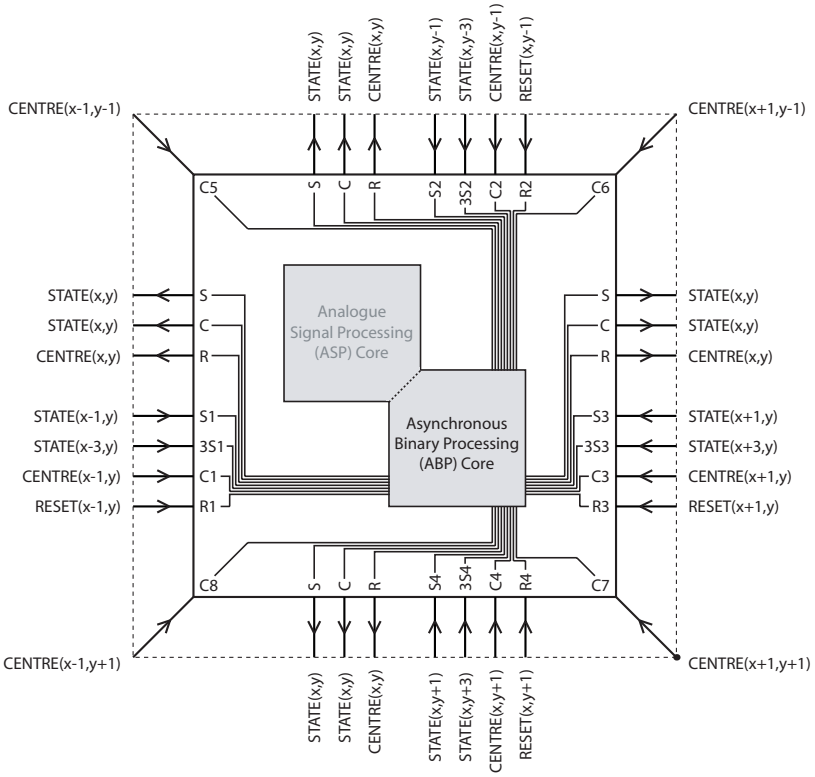


Figure 6.15: Symbol representation of the in-pixel asynchronous binary processing block. Illustrated is the external connectivity of discrete signals, i.e. with other cells, showing the requirements for cellular tessellation. This excludes global control signals and output signals. Nodes have been abbreviated for clarity as follows: S=STATE, R=RESET, C=CENTRE.

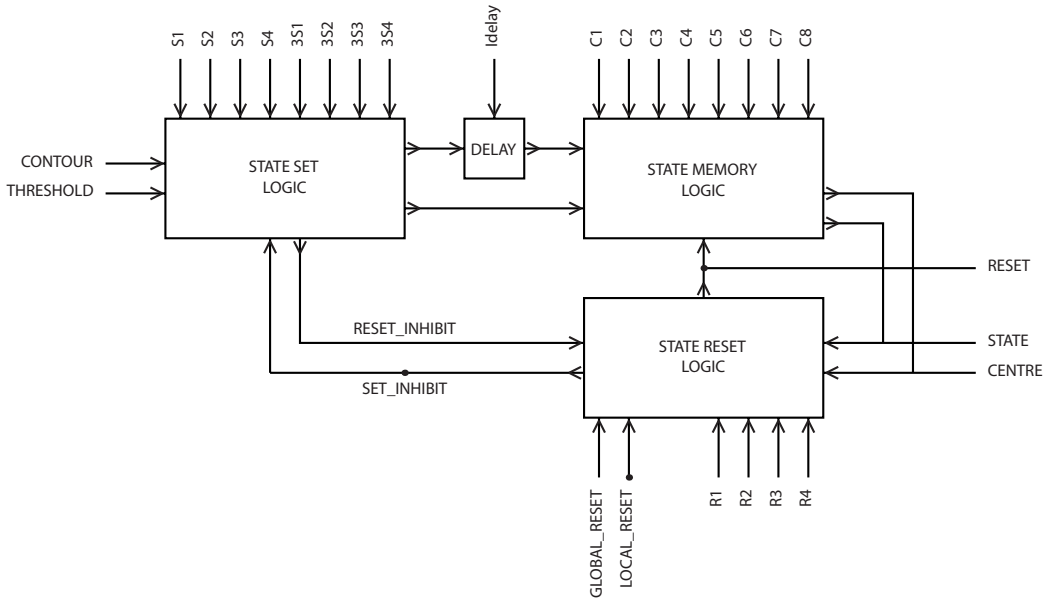


Figure 6.16: Schematic diagram of the Asynchronous Binary Processing (ABP) organisation. Illustrated is the internal connectivity emphasising the signal flow path between the various blocks.

cells (four directly adjacent and four diagonally adjacent) and transmits its CENTRE status back to them.

- Signal feed-through (16 connections): Provides two feed-through connections in each direction (i.e. two leftwards, two upwards, etc) for three-cell proximity state signalling.

Internal

The internal architecture of the ABP core is illustrated in Fig. 6.16.

The ABP core consists of three main (functional) blocks for: (1) state setting (inward propagation), (2) state resetting (back-propagation) (3) state and centroid storage, in addition to some support circuitry.

The state is initially set if a contour is defined (initiation) or if a neighbouring cell signals a state (propagation). A preset delay is added in the propagation path to limit the signalling speed. In parallel the centroid detection checks for a centre-surround condition (i.e. if surrounding cells are set and centre cell is unset). On centroid detection a reset signal is back-propagated through all set cells, until it is blocked by a contour object limit.

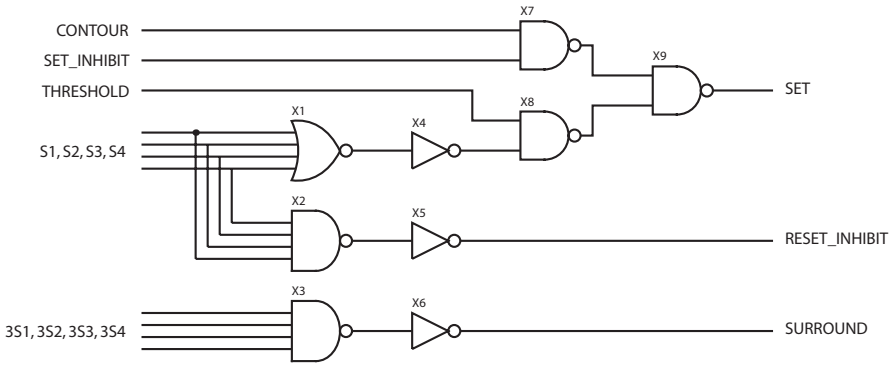


Figure 6.17: Schematic diagram of the state set logic facilitating the forward asynchronous signal propagation.

Furthermore a centroid detection will trigger an off-chip address-event, discussed in detail in the following section.

6.4.2 State Set

The complete combinational logic required to facilitate the state set functionality is illustrated in Fig. 6.17. This block generates the SET, SURROUND and RESET_INHIBIT signals:

- The SET signal is asserted if an adjacent cell has its state set in addition to the THRESHOLD signal being asserted. Alternatively, a SET signal can be generated by a CONTOUR signal provided the SET_INHIBIT condition does not block this. The function of the SET_INHIBIT signal is to avoid oscillation between SET and RESET on completion of a local (object-based) reset cycle.
- The SURROUND signal is generated to assist the centroid detection. The condition for this is if the received surrounding cells (three-pixel proximity) have states set.
- The RESET_INHIBIT signal is generated if any adjacent cells do not have their states set. The purpose for this is described in the following section.

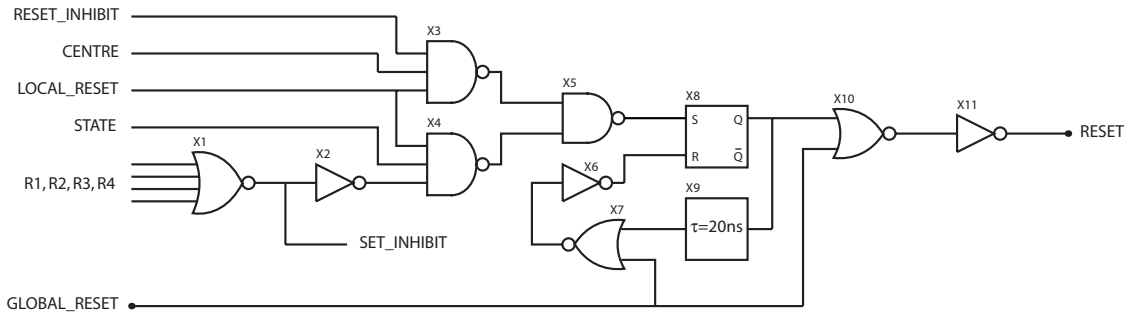


Figure 6.18: Schematic diagram of the state reset logic facilitating the reverse signal (back) propagation.

6.4.3 State Reset

The complete combinational logic required to facilitate the state reset functionality is illustrated in Fig. 6.18. This block generates the RESET and SET_INHIBIT signals:

- The RESET signal is asserted if the LOCAL_RESET mode is enabled and the cell signals a CENTRE (logic X3 and X5). A RESET_INHIBIT signal delays the RESET signal being generated until the inward propagation reaches the central cell, to provide a continuous back-propagation path. Alternatively a RESET signal can be signalled if the STATE is set and any adjacent cell back-propagates a RESET (logic X4 and X5). On generating a RESET signal, a monostable is triggered (logic X6, X8-10) to produce a RESET pulse of sufficient time to reliably back-propagate to adjacent cells.
- The SET_INHIBIT signal (active low) is asserted if any adjacent cells are resetting. This ensures the back-propagation has reliably terminated before a forward propagation can commence.

6.4.4 State Memory

The complete combinational logic required to facilitate the state memory functionality is illustrated in Fig. 6.19. This block completes the asynchronous state machine and provides the STATE and CENTRE signals:

- The STATE signal is latched high on assertion of a SET input and conversely it is latched low on assertion of a RESET input.

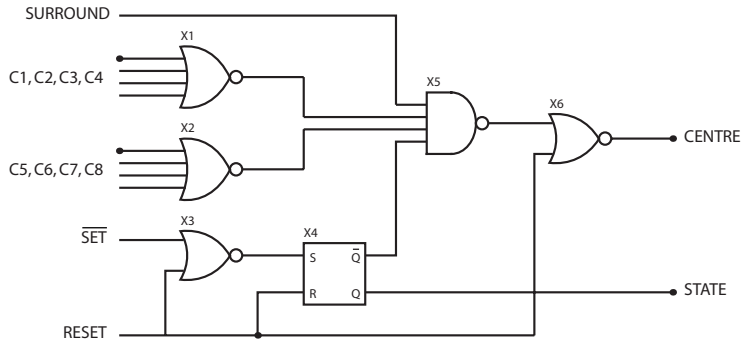


Figure 6.19: Schematic diagram of the state memory logic for centroid determination and state definition.

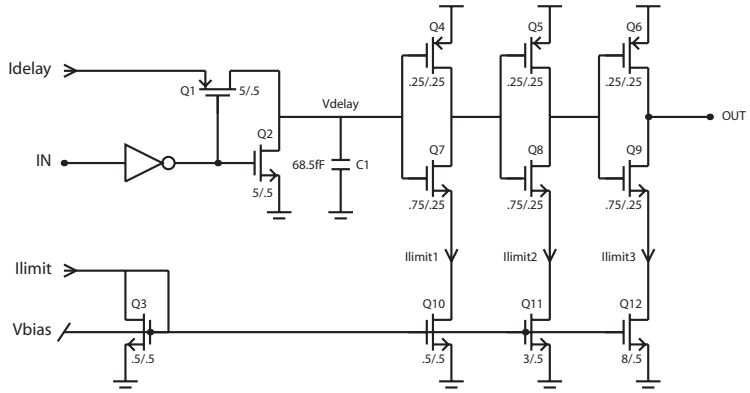


Figure 6.20: Schematic diagram of the current-controlled delay circuit for creating an asynchronous discrete delay.

- The CENTRE signal is asserted when a high SURROUND signal is received in addition to the cells STATE being low. However, this is inhibited if any neighbouring cell (directly adjacent or diagonal) flags a centre. This ensures a single centre is detected in the duration between the CENTRE signal being asserted and the RESET signal being issued.

6.4.5 Delay

As mentioned previously, an artificial delay is inserted in the SET signal path; between the STATE SET logic and the internal STATE MEMORY cell (See Fig. 6.16). The circuit implementation of this delay cell is given in Fig. 6.20.

The delay circuit has a binary input (IN), a binary output (OUT) and two current inputs

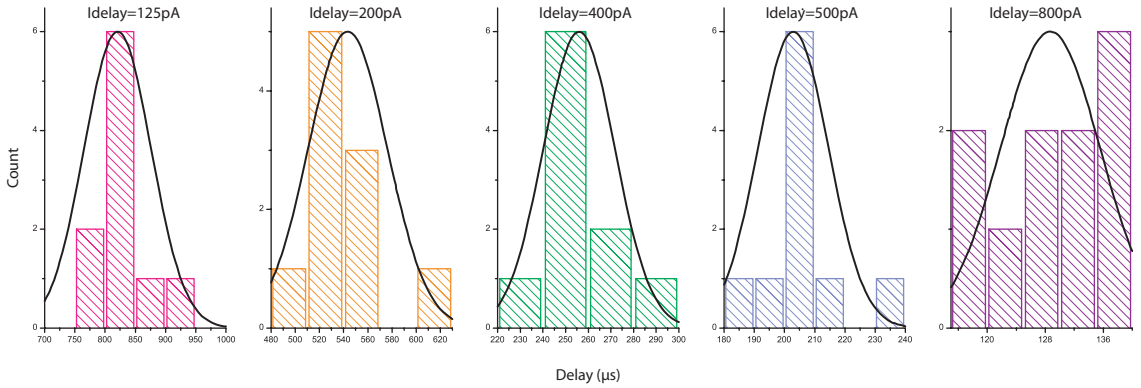


Figure 6.21: Measured delay cell performance for different bias currents (I_{delay}), illustrating statistical variation over a batch of ten samples.

(I_{limit} and I_{delay}). The I_{limit} current defines the current limit on the thresholding inverter circuit, as described previously in Fig.6.13. When the input is high, the I_{delay} current is switched (by device Q1) into a capacitive, integrating node; V_{delay}. A metal-insulator-metal (MIM) capacitor (C1) is used to ensure good linearity and matching. Subsequently, V_{delay} is connected to a three stage NMOS current-limited thresholding inverter; whose threshold sits approximately 300mV below V_{dd}. Therefore for a fixed current, the time delay is given by:

$$\tau_{delay} = \frac{Q_c}{I_{delay}} = \frac{C \cdot V_{threshold}}{I_{delay}} \tag{6.14}$$

Where: τ_{delay} is the time delay, Q_c is the charge stored on the capacitor and $V_{threshold}$ is the threshold voltage of the inverter cascade. For example, an I_{delay} of 1nA would result in a delay of $102\mu s$.

Figure 6.21 presents measured results for the current delay circuit block over a batch of ten fabricated dies. Although a 10-20% spread can be observed, it has to be taken into account that these circuits are collected throughout different wafers and therefore different process corners. Corner simulations demonstrate similar variations in performance. Subsequently, Monte Carlo mismatch simulations suggest that similar delay cells fabricated on the same die; at close proximities will match performance within a 5-10% spread.

6.5 Address Event Representation (AER)

As the described system is both asynchronous and data-driven in nature, it is ideally suited to an event driven output. One such protocol is the Address Event Representation [3]; used extensively in the vision chip arena. The principle behind this data-transmission mechanism is that each pixel has a unique identifier (i.e. its co-ordinate) and when a pixel registers an event this identifier is asserted onto a digital bus. The data is then communicated off-chip through means of an asynchronous handshake.

This section describes the specific AER architecture [8] adopted and the accompanying blocks implemented for off-chip communication.

6.5.1 Architecture

The specific AER architecture implemented is given in Fig.6.22, illustrated for a 4x4 array.

Each pixel in the array has a *sender neuron* that latches a pixels state on an event until the data has been communicated off-chip. The sender neuron initially sends a arbitration request to the row (Y) *arbitration tree*. The role of the arbitration tree is to select a single output in the event of multiple inputs. On selection of a particular row, the *row header* is latched and subsequently the competition passes to the column arbitration tree. A similar process then occurs from the sender neurons to column arbitration tree and back to the column headers until a single column has been latched. On selection of both a row and column, the chip sends a bus request signal off-chip to the receiving device. The address is read off the bus and then a bus acknowledge signal is relayed back to reset the row and column latches that consequently reset the sender neuron state. This selection/arbitration process is then repeated for all events awaiting to be transmitted.

6.5.2 Sender Neuron Circuit

The circuit implementation of the sender neuron block is shown in Fig. 6.23. On a pixel signalling an event, the edge-triggered RS flip-flop (X2) is latched. Subsequently, devices Q1-Q3 are used to divert the output signal flow to negotiate row and then column arbitration. On successful off-chip communication both a row and column reset ($YPIXEL\_RESET$ and $XPIXEL\_RESET$) will be received thus resetting the flip-flop.

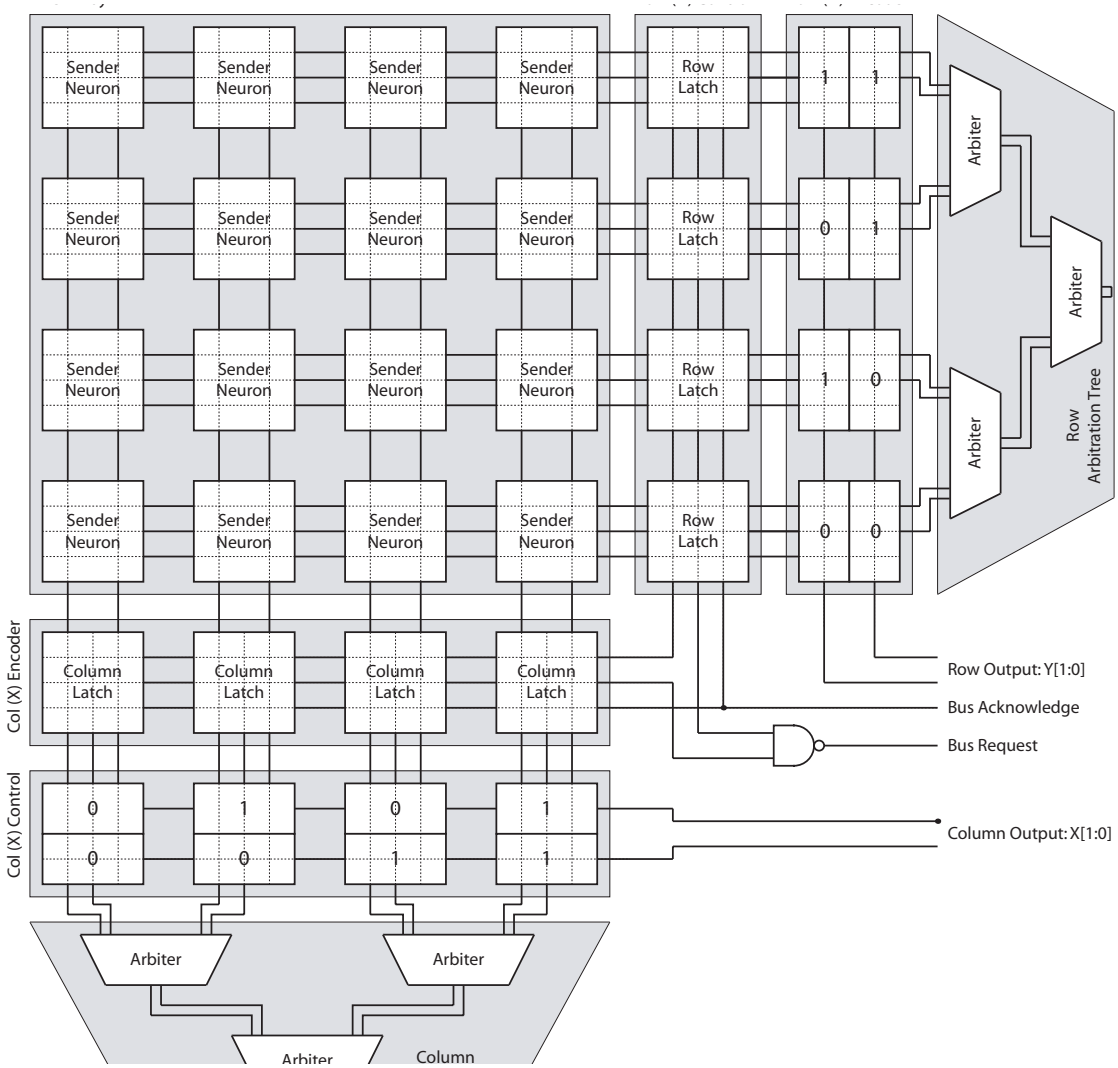


Figure 6.22: Address-Event-Representation (AER) architecture for a 4x4 array. Illustrated are all required sub-blocks for an AER sending device, excluding pull-up and pull-down biases for shared line drivers.

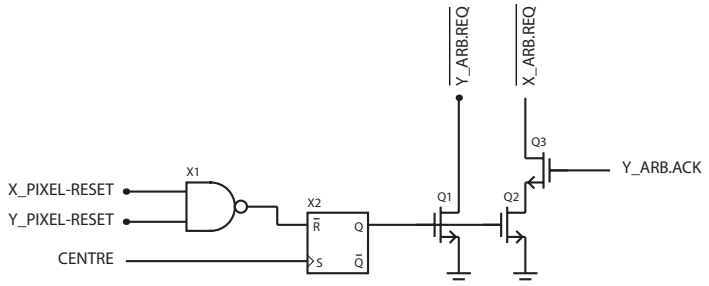


Figure 6.23: Schematic diagram of the sender neuron circuit facilitating the pixel handshake with the AER row/column latches.

The $X\_ARB.REQ$ and $Y\_ARB.REQ$ lines require shared pull-up biasing to avoid floating nodes during low activities.

6.5.3 Column/Row Latch

The circuit implementation of the column/row latch block is illustrated in Fig. 6.24. On a pixel signalling an event, the $\overline{ARB.REQ}$ signal is inverted and passed to the arbitration tree. On arbitration, the $\overline{ARB.ACK}$ of a single row/column latch is asserted that latches the RS flip-flop (X1, X2). Consequently, the $BUS.REQ$ signal is asserted to alert the receiving (off-chip) device that an event is awaiting to be read. In the case of a two-dimensional arbitration tree; as used in ORASIS, the row and column $BUS.REQ$ signals are AND'ed to produce a single chip request. On successful off-chip communication, the receiving device relays a $BUS.ACK$ signal that resets the selected row and column latches and issues the $PIXEL\_RESET$ signals to reset the sender neuron within the sending pixel.

The $BUS.REQ$ line requires a shared pull-down bias to avoid a floating node during low activities.

6.5.4 Arbiter Circuit

The circuit implementation of the arbiter block; used to synthesise the row and column arbitration trees is illustrated in Fig. 6.25. The arbitration tree has the task to select one of many requests, facilitated through a binary tree hierarchy. The arbiter cell operates on a single input pair, i.e. by selecting one of two outputs, resolving contention by using a high gain positive feedback element. The arbiter operates as follows:

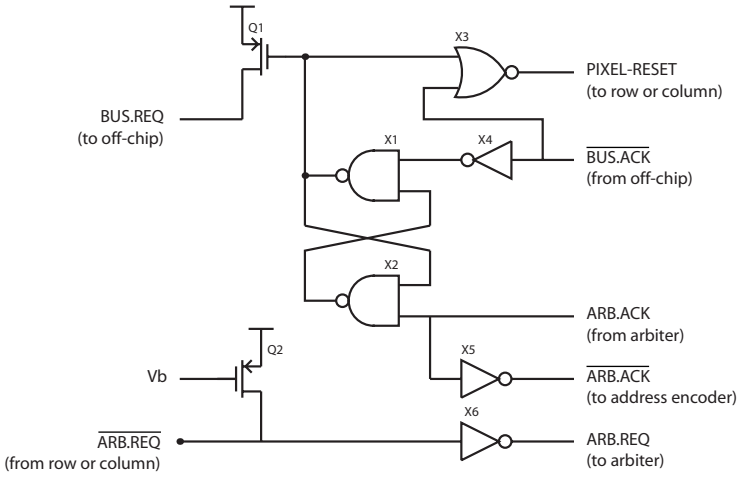


Figure 6.24: Schematic diagram of the row and column latch circuit; locking a pixel's address upon arbitration until being successfully transmitted off-chip.

- If $ARB_1.REQ$ and/or $ARB_2.REQ$ is asserted, $ARB.REQ$ is asserted to a higher arbitration level (logic OR operation).
- If either $ARB_1.REQ$ or $ARB_2.REQ$ is asserted, the RS flip-flop (X2, X3) is steered accordingly and on the arbiter receiving an $ARB.ACK$ signal from a higher level, it signals an $ARB_X.ACK$ to the requesting branch.
- If neither $ARB_1.REQ$ nor $ARB_2.REQ$ are asserted the RS flip-flop enters an undefined state, however this doesn't effect the operation since it will not pass an $ARB.REQ$ signal to a higher level.
- If both $ARB_1.REQ$ and $ARB_2.REQ$ are asserted, the RS flip-flop selects the last asserted and on receiving an $ARB.ACK$ signal from a higher level, it signals an $ARB_X.ACK$ to the selected branch.

6.5.5 Address Encoder

The address encoder circuit is based on a *wired-OR* topology, shown in Fig.6.26. This implementation is both reliable and effective for encoding the output, as the arbitration tree can only select a single output. Therefore each output can be *hard-wired* to assert the required digital representation on the AER bus.

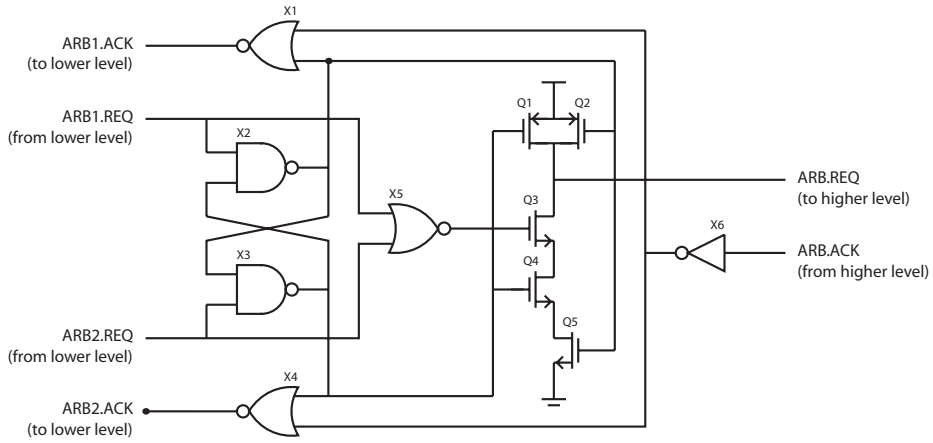


Figure 6.25: Schematic diagram of the arbiter circuit, interconnected hierarchically to synthesise the arbitration trees for row and column selection.

One pull-down bias is required per bus line (bit), to prevent the bus output from floating during low activities.

6.6 Fabricated prototypes

The proposed system was developed and fabricated in two stages: ORASIS-P1; the test chip and ORASIS-P2; the full system (48x48 array). These integrated circuits are implemented in a standard, commercially available CMOS process; UMC 0.18μm single-poly, six-metal layer, triple-well (MM/RF) technology.

6.6.1 ORASIS-P1

The first fabricated circuit was developed to demonstrate the feasibility of the proposed distributed analogue signal processing (ASP) and asynchronous binary processing (ABP) architectures. A 4x8 element ASP and 1x15 element ABP were separately implemented to verify the expected operation of the two custom (distributed) processing cores. Furthermore, several test structures and prototype circuits were included for validation and characterisation. The layout of this this test chip (ORASIS-P1) is illustrated in Fig. 6.27. Details of the test platform used to verify this IC are given in Appendix D.

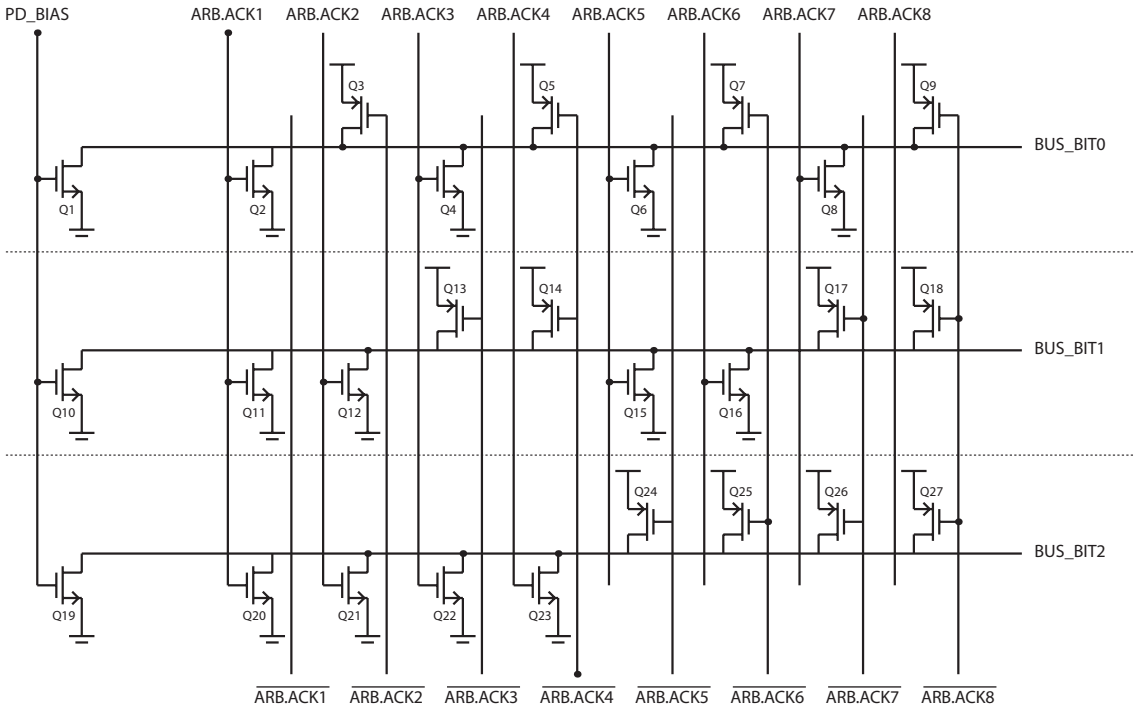
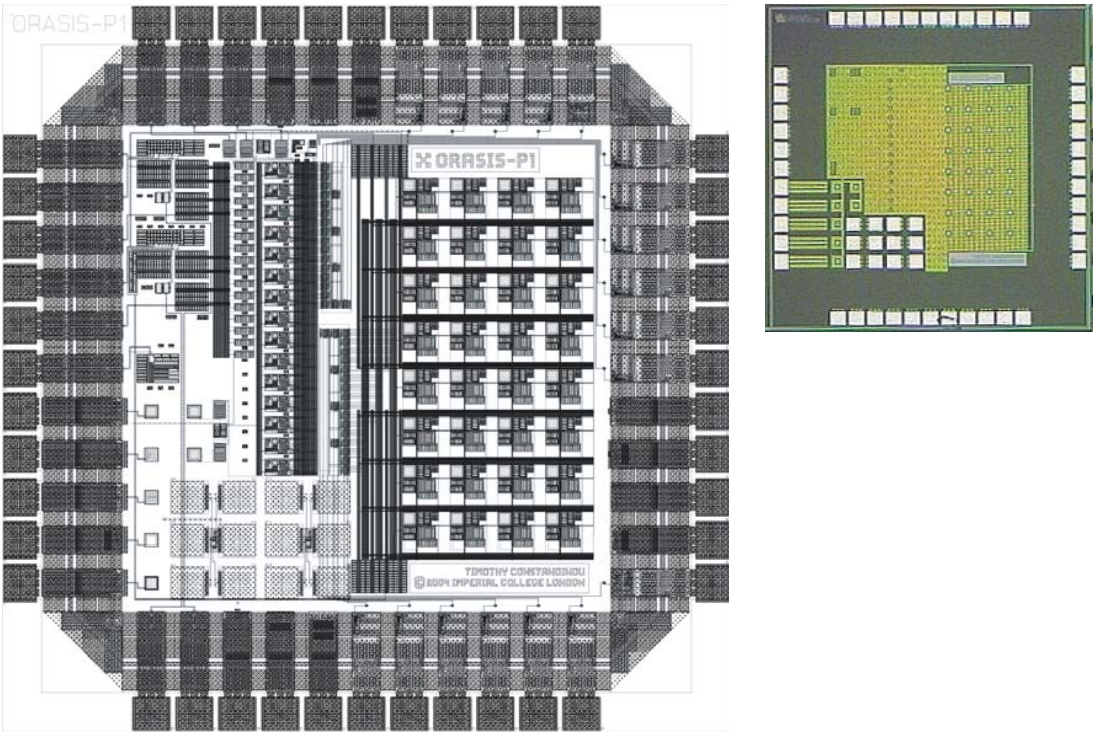


Figure 6.26: Schematic diagram of the address encoder circuit, shown for an eight input, 3-bit output example.



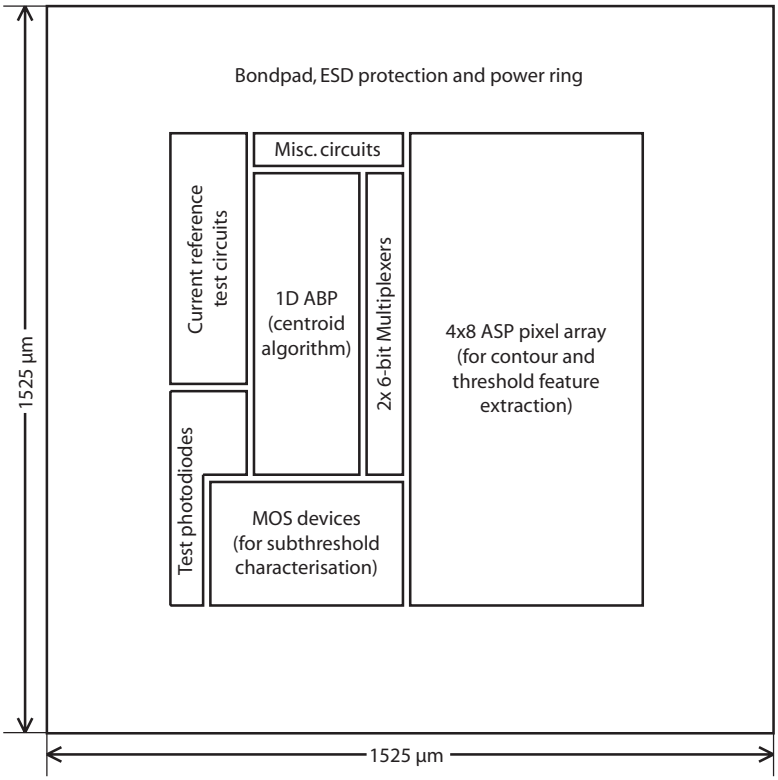


Figure 6.27: The ORASIS-P1 test chip layout (top), microphotograph (top right) and basic floorplan (bottom); implemented in UMC 0.18μm 1P6M mixed-mode CMOS, accessed through Europractice (IMEC). The die size is 1.525mm x 1.525mm (excluding scribe line). Metal layers 5 and 6 have been excluded for clarity.

6.6.2 ORASIS-P2

The complete system (ORASIS-P2) layout is given in Fig. 6.28.

The design uses 84 physical bondpads (17 per side) uniformly distributed (at $200\mu m$ spacing) with $75\mu m \times 65\mu m$ passivation opening for bonding within a J-leaded chip carrier package (PLCC84). The padring is constructed from a standard cell library provided by virtual silicon for $60\mu m$ inline pads. As the design uses RF top metal (20KA) option, the 5 metal layer (logic process) cell library was used to avoid design rule violations. The padring is split at the left and right edges; with top part being the analogue section and bottom part being the digital section.

The master current references are on the top edge using off-chip resistors for wide tunability and fine adjustment to compensate for process variation. Four master currents per reference are sourced to the array corners to drive the current distribution network (see Fig. 6.29). The buffers illustrated provide the digital fanout for the globally distributed control inputs.

The address event representation hardware is situated at the bottom (column control) and right (row control) of the array. The pixel array implemented is a 48x48 matrix using a tessellation of nine different cell types (edge, corner and regular) for correct termination to provide good array utilisation. The regular cell implementation (layout) is given in Fig. 6.30.

The pixel floorplan is arranged such that the ASP (approximately top 65% area) is separate from the ABP (approximately bottom 35% area). The distributed data "bus" is routed in metals layers 1 and 3 vertically along the left pixel edge and metal layers 2 and 4 horizontally along the bottom pixel edge (See Table 6.1). Metal layer 5 is used for current and power supply distribution (horizontal), from the left and right current distribution with a break after the central pixel (X24) column. Metal 6 is used as a ground plane and a light blocking screen, to minimise photo-absorption in the substrate, apart from at photodiode openings.

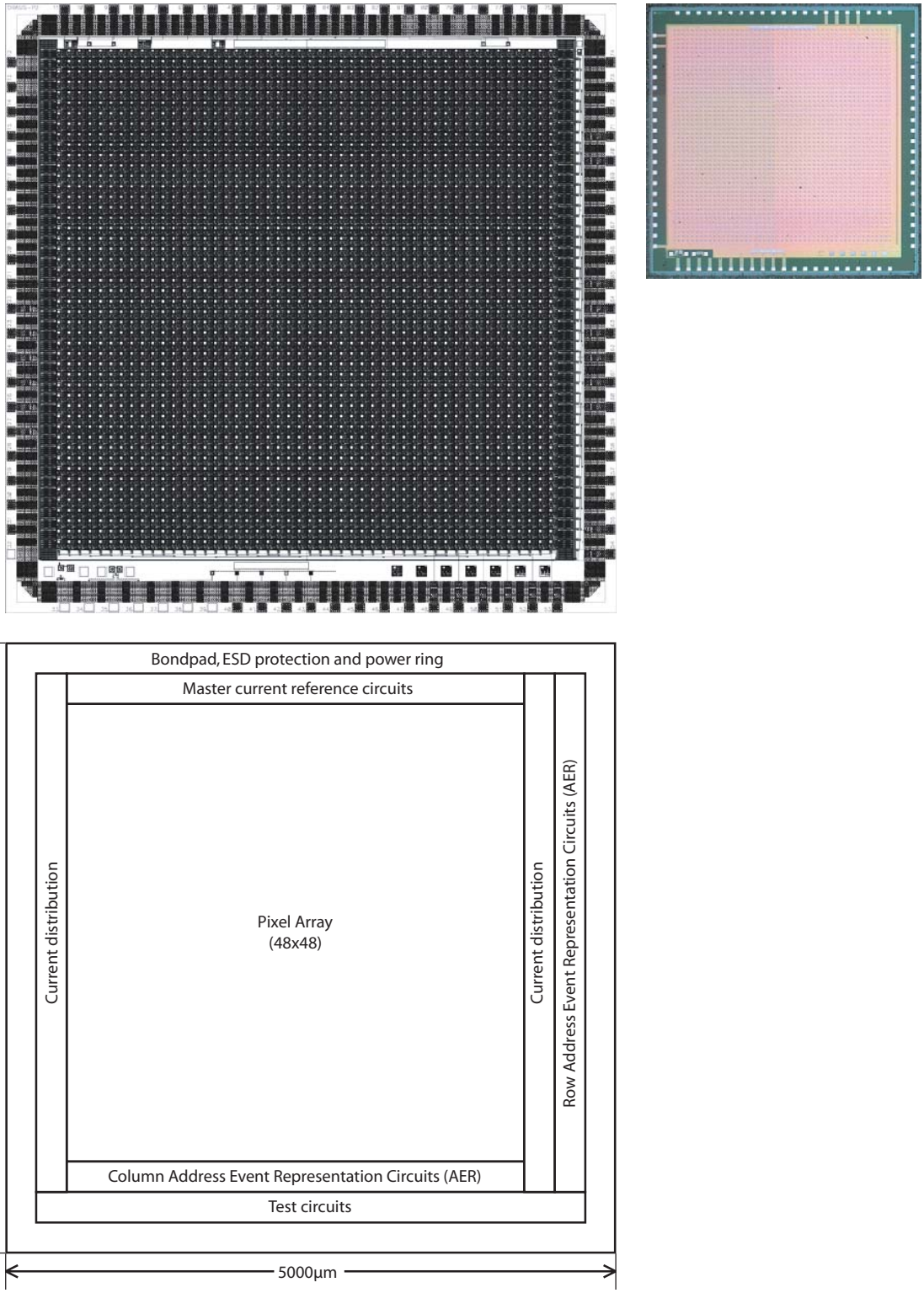


Figure 6.28: The ORASIS-P2 chip layout (top), microphotograph (top right) and basic floorplan (bottom); implemented in UMC 0.18μm 1P6M mixed-mode CMOS, accessed through Europractice (IMEC). The die size is 5.0mm x 5.0mm (excluding scribe line). Metal layers 5 and 6 have been excluded for clarity.

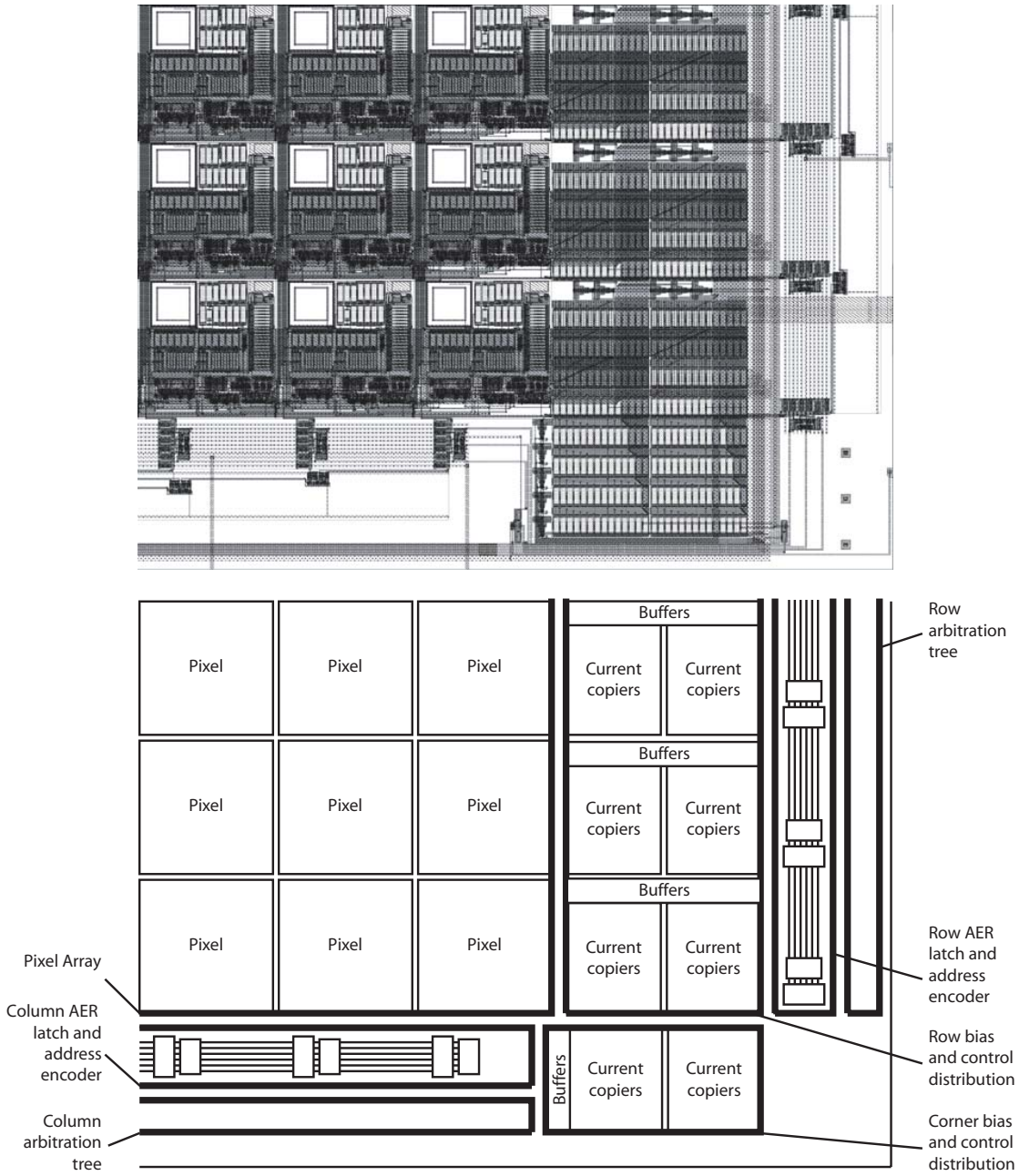


Figure 6.29: The bottom-right array corner layout (top) and floorplan (bottom), illustrating the implementation of the current distribution scheme and array-side address-event circuitry. Metal layer 6 has been excluded for clarity.

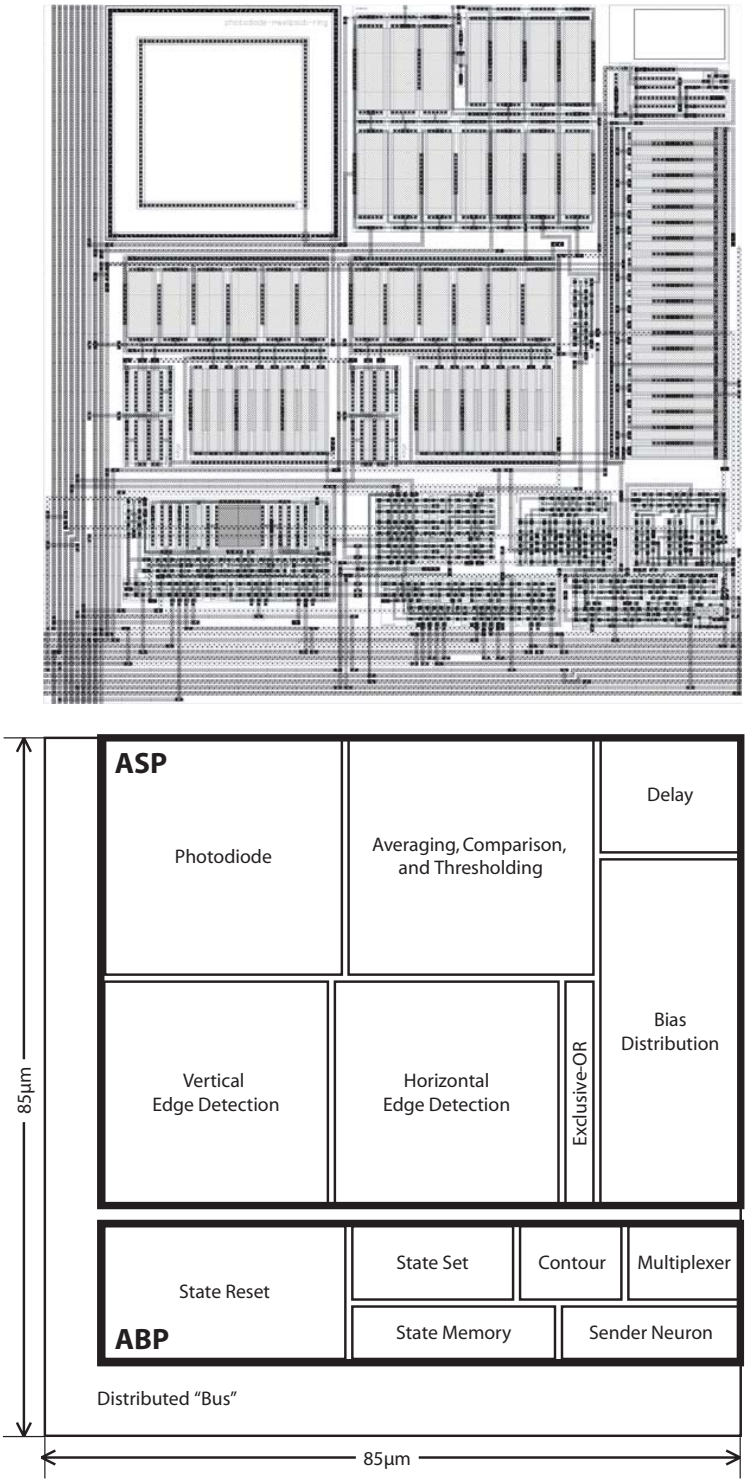


Figure 6.30: The ORASIS-P2 regular cell layout (top) and floorplan (bottom). The cell size is $85\mu m \times 85\mu m$ with $30\mu m \times 30\mu m$ active photodiode area, giving a 12.5% surface fill factor. Metal layers 5 and 6 have been excluded for clarity.

	Metal 1 (top to bottom)		Metal 2 (left to right)		Metal 3 (top to bottom)		Metal 4 (left to right)		Metal 5 (left to right)	
	↑	↓	←	→	↑	↓	←	→	←	→
1	R1	R	IOUT1	ILOC	C1	C	VOUT	VIN1	IB[23:0]	IB[23:0]
2	R	R3	ITHRU	ILOC	C	C3	EOUT1	EIN1	IT[23:0]	IT[23:0]
3	3S1	2S1	R4	R	S	S3	C4	C	VDDA	VDDA
4	2S1	S1	R	R2	S3	2S3	C	C2	VSSA	VSSA
5	S1	S	C5	C1	2S3	3S3	C7	C3	VSS	VSS
6	RESX	RESX	C1	C6	$\overline{REQX}$	$\overline{REQX}$	C3	C8	VDD	VDD
7	IOUT3	ITHRU	3S4	2S4	EOUT2	EIN2	S	S2	OUT1	OUT1
8	IOUT2	ILOC	2S4	S4	VOUT	VIN2	S2	2S2	OUT0	OUT0
9	ICOL	ICOL	S4	S	-	-	2S2	3S2	MODE	MODE
10	-	-	RESY	RESY	-	-	-	-	RESL	RESL
11	-	-	ACKY	ACKY	-	-	$\overline{REQY}$	$\overline{REQY}$	RESG	RESG

Table 6.1: Tessellating cellular interconnectivity; in total, each cell has 190 connections with adjacent cells.

6.7 System Results (Simulated)

System verification has been divided into four sections; partly to reduce simulator load and partly to obtain comprehensive and standalone results at each stage. The first three sections deal with scaled-down ASP, ABP and AER architectures individually, with the final section presenting the overall system results. All the test schematics used are provided in Appendix C.

6.7.1 ASP

The ASP core was verified by simulating a 16×16 array with a static single-object image (photocurrent array) hardwired under different configurations. Each system state is tested through all process corners to verify robustness to process variations. Furthermore, the total ASP power consumption is measured under each test condition, given in table 6.2.

The analogue power consumption contribution can be relatively accurately calculated

Corner	tt	ss	ff	snfp	fnsp
State 1: $V_{THRES\_MODE} = 0$, $V_{GLOBAL\_AV} = 1.8$, $I_{photo} = 300p$, $I_{photoObj} = 50p$, $I_{photoAverage} = 228p$, $I_{bias} = 1n$, $I_{tune} = 250p$, $I_{global} = 0$					
Analogue	$4345nW$	$4349nW$	$4340nW$	$4343nW$	$4349nW$
Core digital	$1771nW$	$4018nW$	$1547nW$	$1736nW$	$2272nW$
Total ASP power (16×16)	$6116nW$	$8367nW$	$5887nW$	$6079nW$	$6621nW$
Average ASP power (per cell)	$23.89nW$	$32.68nW$	$23.00nW$	$23.75nW$	$25.86nW$
State 2: $V_{THRES\_MODE} = 1.8$, $V_{GLOBAL\_AV} = 1.8$, $I_{photo} = 100p$, $I_{photoObj} = 300p$, $I_{photoAverage} = 153p$, $I_{bias} = 1n$, $I_{tune} = 250p$, $I_{global} = 0$					
Analogue	$4081nW$	$4077nW$	$4082nW$	$4079nW$	$4082nW$
Core digital	$1646nW$	$1463nW$	$3382nW$	$1650nW$	$1998nW$
Total ASP power (16×16)	$5727nW$	$5540nW$	$7464nW$	$5729nW$	$6080nW$
Average ASP power (per cell)	$22.37nW$	$21.64nW$	$29.16nW$	$22.38nW$	$23.75nW$
State 3: $V_{THRES\_MODE} = 1.8$, $V_{GLOBAL\_AV} = 1.8$, $I_{photo} = 1n$, $I_{photoObj} = 100p$, $I_{photoAverage} = 761p$, $I_{bias} = 1n$, $I_{tune} = 500p$, $I_{global} = 0$					
Analogue	$6356nW$	$6340nW$	$6374nW$	$6354nW$	$6358nW$
Core digital	$1777nW$	$1551nW$	$4016nW$	$1745nW$	$2272nW$
Total ASP power (16×16)	$8133nW$	$7840nW$	$10390nW$	$8099nW$	$8629nW$
Average ASP power (per cell)	$31.77nW$	$30.82nW$	$40.58nW$	$31.64nW$	$33.71nW$

Table 6.2: Simulation results for a 16×16 ASP core indicating average power consumption levels for typical stimuli through the different process corners.

due to the fact that all biasing is current-input, i.e. current-mode. Therefore the average cellular (analogue) current consumption is expressed in Eq. 6.15. As expected and confirmed by the corner simulation results, this analogue contribution is nearly constant through the different process corners.

$$P_{ASP(ana)} \approx V_{dda} \cdot (7.5I_{bias} + 3I_{tune} + 7I_{photoAverage}) \tag{6.15}$$

However the core digital power consumption is observed to vary substantially with process variation. This is due to also supplying the current-limiting threshold detectors that experience a shift in region of operation. The core digital supply current is therefore *independent* of average photocurrent level and ASP configuration. The maximum limit of digital (core) current consumption within the ASP core is expressed in Eq.6.16. This is based on the maximum simulated current consumption for typical values of internal mismatch within three current-limiting threshold detectors.

$$P_{ASP(dig)}(max) \approx V_{dd} \cdot (10I_{limit}) \tag{6.16}$$

Where: I_{limit} is the first stage current-limit set in the thresholding inverter.

As this presented ASP architecture (with exception of bias distribution) requires no off-array circuits, the total power consumption can be reduced to a cellular average. Based on the simulated ASP results and measured photodiode responsivity this average is expected to be in the range 15-40nW per cell (pixel). Therefore scaled to a megapixel array, this would give 15-40mW total power consumption for both phototransduction and front-end binary feature extraction.

6.7.2 ABP

The ABP core was verified by simulating a 9×9 array with a static single-object image hardwired; by means of providing the ASP outputs (contour and threshold) as a matrix of distributed inputs within the ABP array. For $I_{delay} = 1.3nA$, the transient behaviour is illustrated in the simulation results given in Fig. 6.31.

These results illustrate the algorithms *bio-pulsating* action distributed through internal (array) memory for a preset circular object. The cellular STATE can be observed to fill

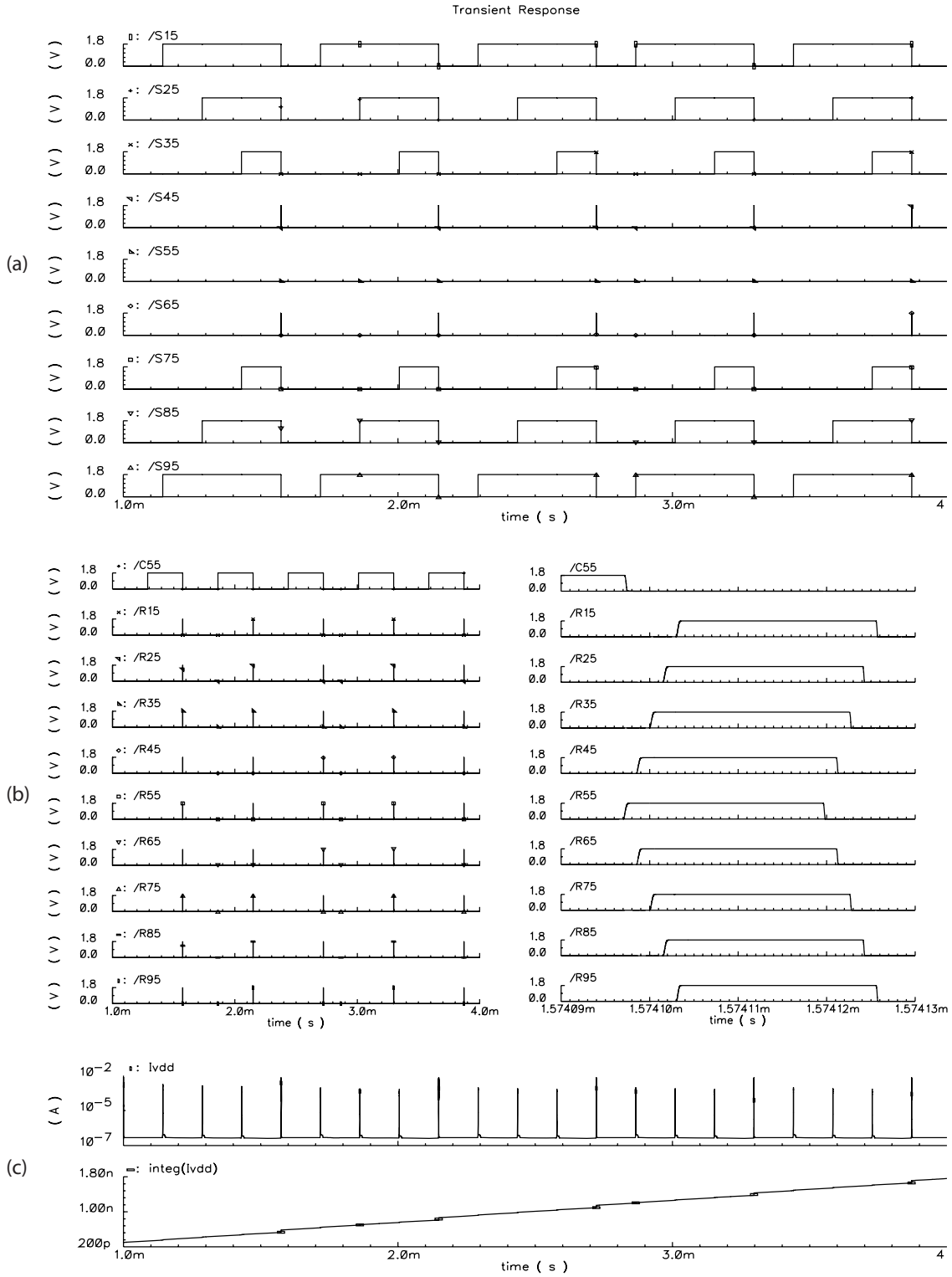


Figure 6.31: Transient analysis simulation results for a 9×9 ABP core illustrating bio-pulsating action for a single object image. Results shown are taken across the central row (Y=5) for: (a) state propagation, (b) reset back-propagation and (c) current consumption.

inwards (Fig. 6.31(a)). On convergence to a *centroid* cell, the CENTRE signal is flagged, causing a RESET back-propagation (Fig. 6.31(b)). This sequence then reinitialises and thus for a static input, the process repeats in a periodic manner.

The *average* power consumption determined in this simulation is 470nW, providing 1739 processed (centroid) results per second. In a 9×9 array with a circular object of diameter 7 pixels, the active pixels are: πr^2=38.5 pixels. By assuming static power dissipation in non-active cells to be negligible (reviewed later), the ABP consumption per active pixel is therefore: 470n/38.5=12.21nW. Furthermore, as the delay constant scales linearly with I_{delay}, both the activity and power consumption also scale linearly.

6.7.3 AER

The AER architecture was tested using a 12×12 sender neuron array with 12-input row/column latches, encoders and arbitration trees. Multiple sender neurons selected at random positions were programmed to output colliding events to test robust arbitration and bus selection. The colliding events have been arranged in two phases; the sender neurons at positions: (2,9) (6,9) (5,10) (6,10) (9,10) signalling events at t=50μs and sender neurons: (4,2) (8,2) (1,3) (5,3) (11,4) (12,4) (4,7) (8,7) (1,8) (5,8) (11,8) (12,8) (9,9) (12,9) (1,10) (2,10) (2,11) (6,11) (9,11) outputting at t=80μs. The system REQ and ACK signals facilitating the off-chip handshake are connected directly through a 3μs delay; representing the receiving device. The results are given in Fig. 6.32.

The current consumption profile (Fig. 6.32(b)) suggests that a unit energy is required per event in addition to a static dissipation proportional to the number of rows and columns. This is expressed in Eq. 6.17.

$$P_{aer} \approx V_{dd} \cdot (N_{rows} + N_{columns}) \left[I_{static} + (I_{eventAv} \cdot N_{events} \cdot t_{eventAv})\right] \tag{6.17}$$

Where: I_{static} is the static current (per row/column header overhead), N_{rows} and $N_{columns}$ are the number of rows and columns respectively, $I_{eventAv}$ is the average event current, N_{events} is the number of events (per second) and $t_{eventAv}$ is the average event time.

The value of I_{static} is proportional to the bias current; used in the address encoder pull-down, bus request pull-up, etc. Thus a higher bias can increase the throughput of the address-event bus at the expense of static dissipation. From the simulated results presented

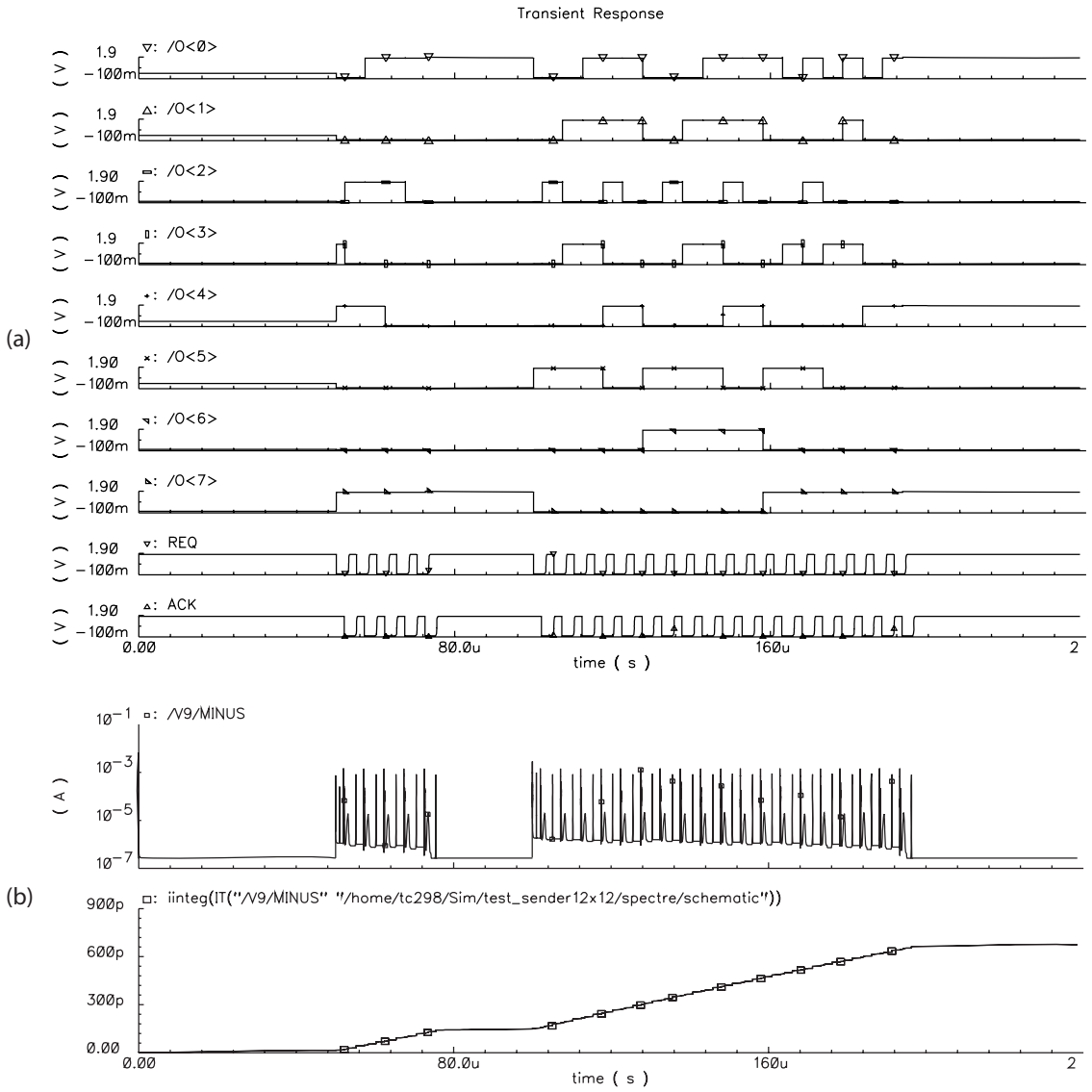


Figure 6.32: Transient analysis simulation results for a 12×12 AER sending architecture illustrating arbitration for 24 colliding events. Results shown are: (a) the AER bus output/handshake and (b) current consumption.

in Fig. 6.32, typical AER consumption values (for $I_{bias} = 100nA$) are: $I_{static} = 11.16nA$ and $I_{eventAv} = 225.7nA$ (determined from $E_{eventAv} = 27.08pJ$). For example, in a 12×12 array, outputting 10K events per second the total AER consumption would be 752.9nW.

6.7.4 Overall System

The overall system was simulated and therefore verified in two stages. Initially a complete distributed array is simulated with ideal current and voltage sources to confirm correct array processing functionality and thus validate the cellular processing element. The second stage involves hierarchically arranging the array to include the current bias and control distribution tree. Thus the second stage simulation is intended to truly represent a scaled final system validation.

Complete Array

The combined operation of the distributed ASP and ABP cores with the AER off-chip communication is verified by testing a complete 12×12 array. The simulation results are given in Fig. 6.33.

The total array power consumption is consistent with the expected level; derived from appropriately scaling constituent component requirements. This comparison is given in Table. 6.3. The small difference in between estimated (4350.2nW) and simulated (4816.4nW) results can be attributed to the static power dissipation in the ABP core (this was previously assumed negligible). This therefore represents a static dissipation within the ABP core of 3.24nW/cell.

Complete System

Having verified correct operation in the distributed array, a scaled-down system mock is tested to validate correct hierarchy and to determine power consumption overhead in global bias distribution. A 12×12 array, arranged in exactly the same hierarchy (i.e. system/array/corner/row/cell) as the final chip (48×48 array). The simulation results are given in Fig. 6.34.

As expected, the core power (vdd) consumption is in line with that shown previously, i.e. as in the 12×12 array simulated results. The power consumption due to global bias

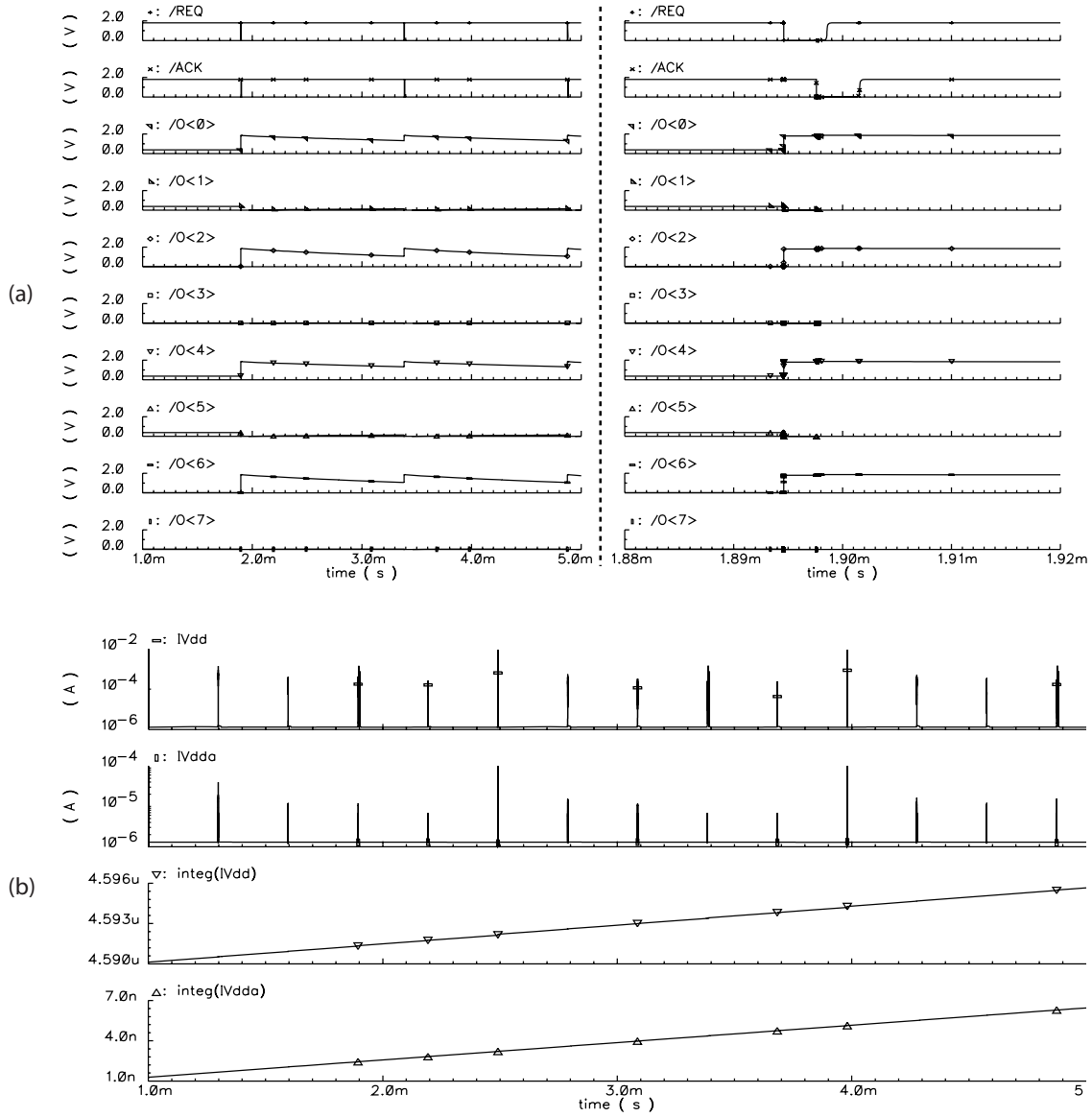


Figure 6.33: Transient analysis simulation results for a 12×12 complete array for a single circular object input (8 pixel diameter). Results shown are: (a) the AER bus output/handshake; event at position (6,6) and (b) current consumption.

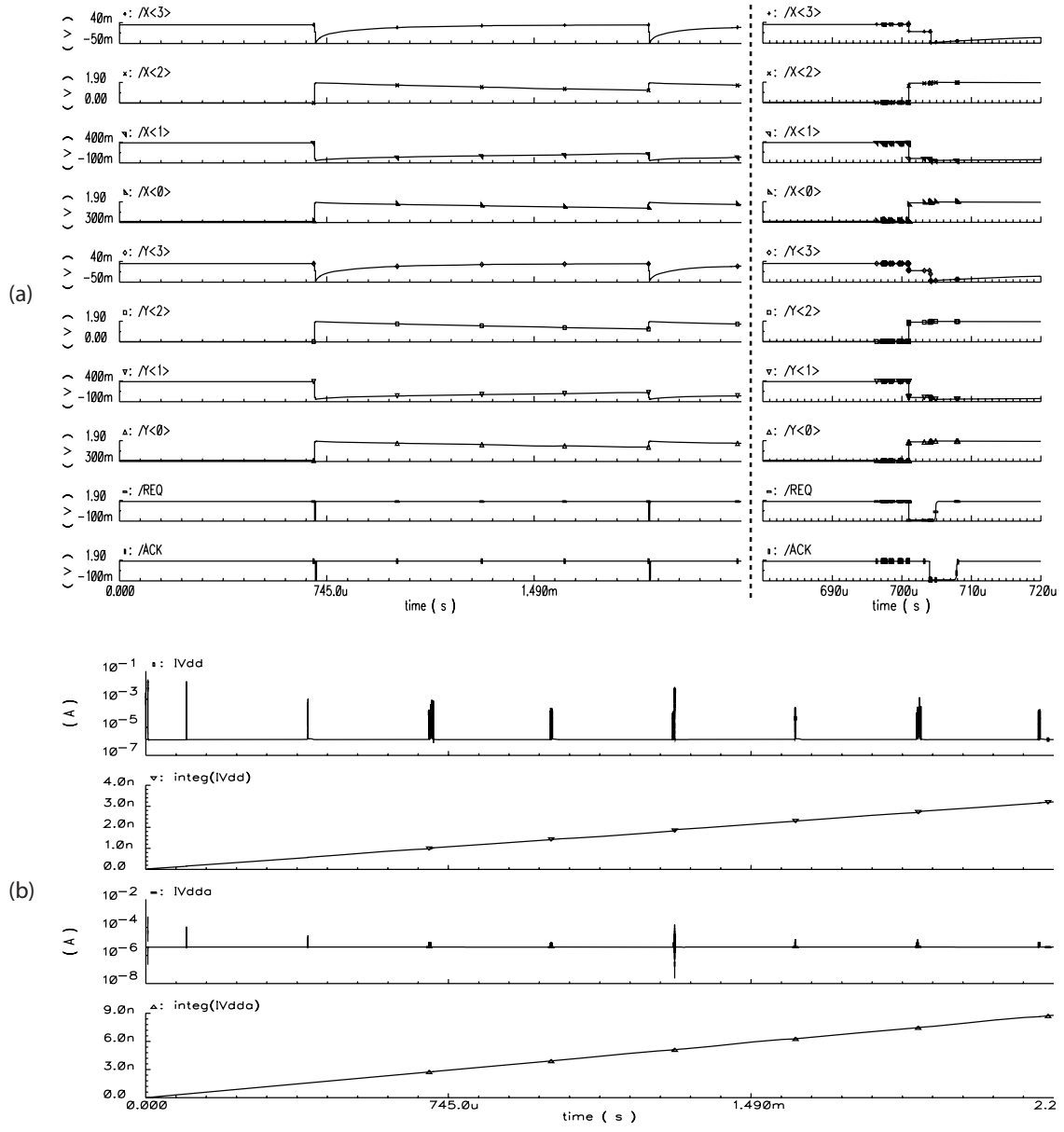


Figure 6.34: Transient analysis simulation results for a 12×12 complete system (including bias and signal distribution) for a single circular object input (8 pixel diameter). Results shown are: (a) the AER bus output/handshake; event at position (6,6) and (b) current consumption.

	Cellular Power		Active Cells		Distributed Power		
Component	Static	Dynamic	Static	Dynamic	Static	Dynamic	Total
ASP	23.89nW	-	144	-	3440nW	-	3440nW
ABP	-	12.21nW	-	50[1]	-	610.5nW[1]	610.5nW
AER	11.16nW	48.75pJ[2]	24[3]	1[4]	267.8nW[3]	31.68nW[4]	340.9nW
Array (estimated)	-	-	-	-	3708nW	642.2nW	4350.2nW
Array (simulated)	-	-	-	-	-	-	4816.4nW

[1] Assuming an input image including a circular object of 4 pixel radius.
[2] Energy required per address event output.
[3] Static dissipation is per row and column header.
[4] Assuming each active cell to be a centroid; generating approximately 650 events per second.

Table 6.3: Comparison between expected (based on constituent ASP, ABP, AER embedded arrays) and simulated power consumption for a combined 12×12 array.

distribution is observable as the difference in analogue power (vdda) consumption between the simulated array and system results. Furthermore, it is apparent that the static (leakage) dissipation is substantial and in fact is the main source of power consumption within the ABP core; even at high activities.

6.8 System Results (Measured)

6.8.1 Test Method

A custom testboard has been developed for verifying system functionality (full schematic provided in Appendix D). The approach taken is to have a dedicated microcontroller (Microchip PIC18LF4620) facilitating the address event handshake and subsequently storing the address event data into internal memory until filled, then streaming out to a PC via a standard UART (RS232) interface. The drawback of this approach is that the test chip is only tested in short bursts and therefore the output data (although processed in realtime), is only available off-line. This is due to the limited bandwidth of the UART (a maximum of 115200kbps). The source code for the address-event handshake and sampling has also been included in Appendix B.

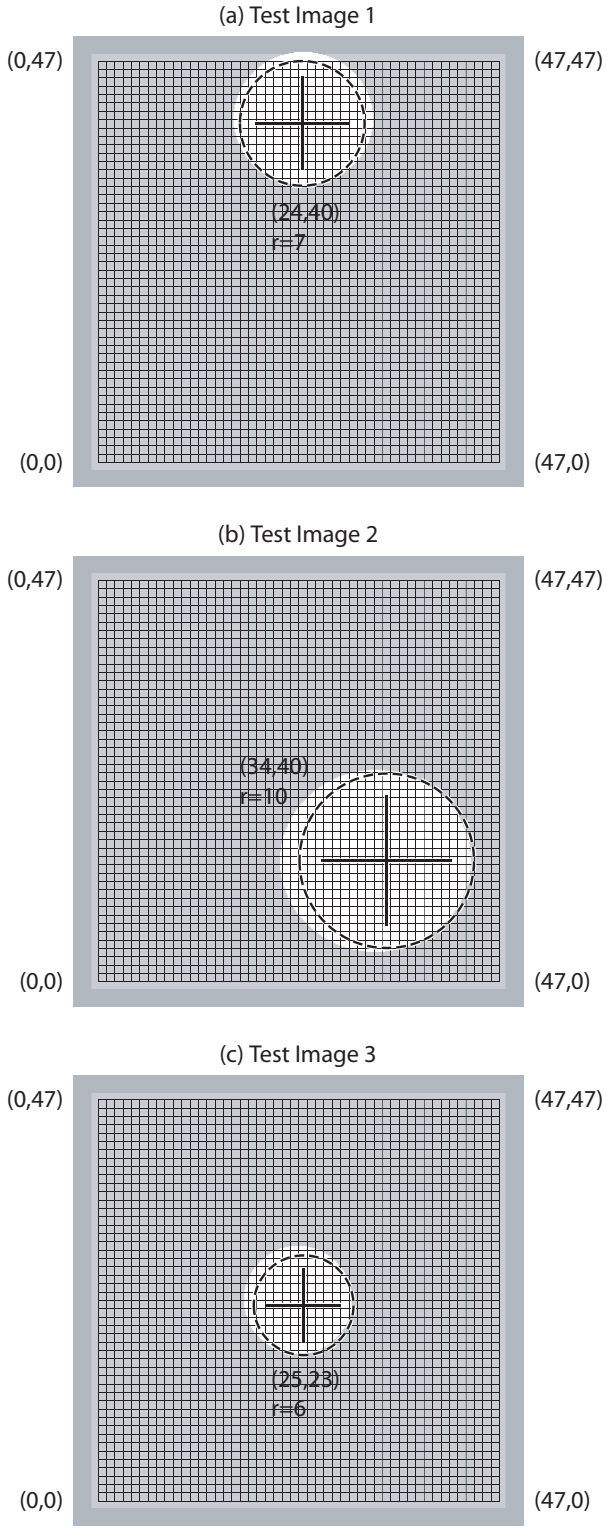


Figure 6.35: Test images with single uniform objects, with pixel grid overlayed including measured centroid position and size.

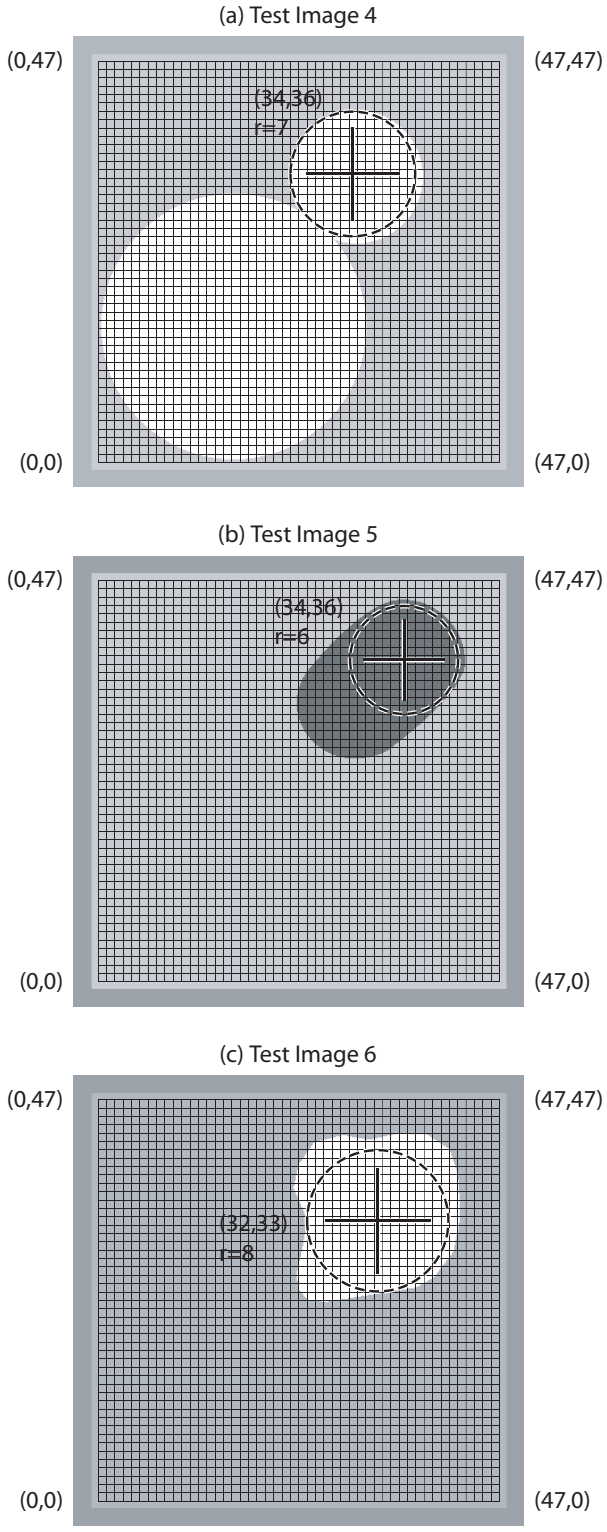


Figure 6.36: Test images with single non-uniform objects, with pixel grid overlayed including measured centroid position and size.

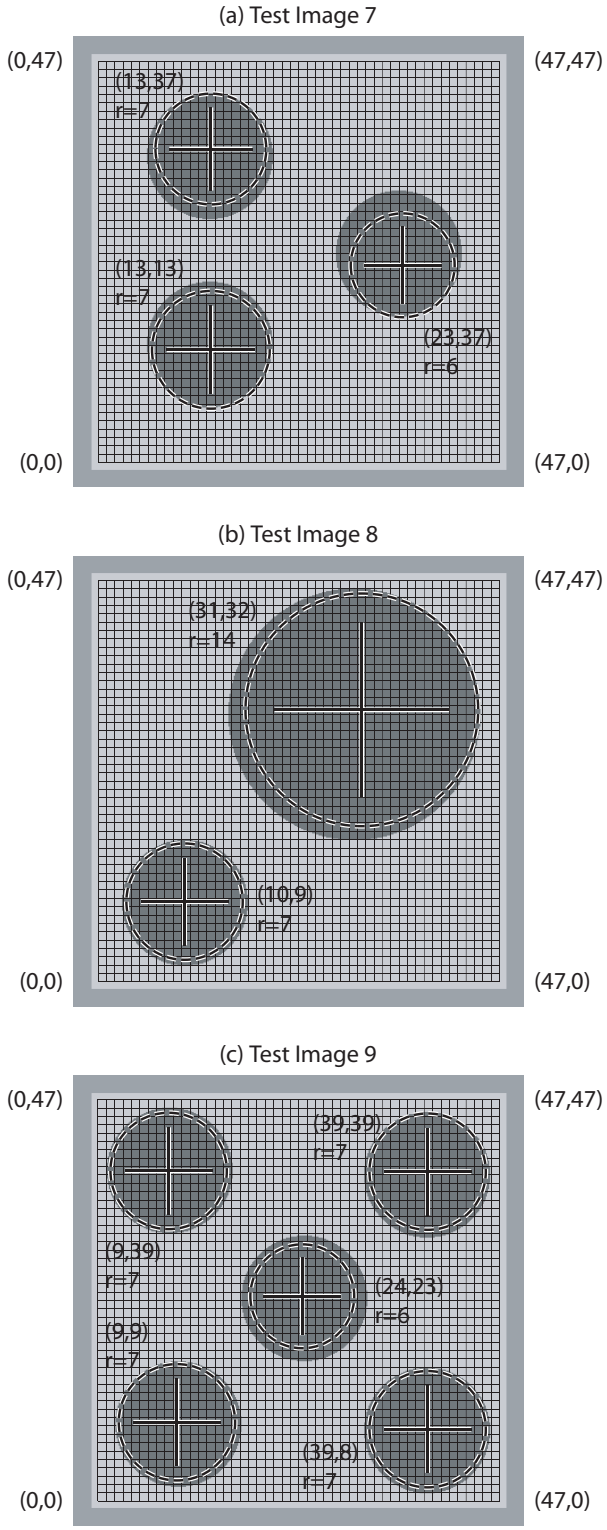


Figure 6.37: Test images with multiple uniform objects, with pixel grid overlayed including measured centroid positions and sizes.

For image acquisition, a 2/3" format CCTV lens (Pentax C1614A) is mounted a fixed distance (16mm fixed focal length) above the bare silicon surface. Subsequently, a thin-film transistor (TFT) liquid crystal display (LCD) is used to produce the image, positioned approximately 40cm perpendicular to the focal plane of the ORASIS_P2 chip. The region on the TFT display focused onto the photodiode array is then determined through power consumption measurements. Initially, a narrow white rectangular region incident inside the photodiode array is extended both in X and Y axis, until no further increase in current consumption is measurable. At this point the entire array has been illuminated and therefore the array boundaries have been established. Furthermore, to determine the incident light intensity, test photodiodes devices previously characterised are now used to provide this calibration.

6.8.2 System Functionality

This setup is used to confirm system functionality within the intended design specifications. Sample images, projected onto the ORASIS_P2 chip and corresponding measurements are presented in Figs. 6.35, 6.36 and 6.37. These illustrate both single and multiple object detection, centroiding and sizing. Typically the measured centroid and size measurements are within the actual object boundaries, i.e. the system tends to under rather than over estimate. Furthermore, uneven objects are successfully detected but with inaccurate centroid and position estimates, again within the actual object boundaries (see Fig. 6.36b,c). However, overlapping objects are detected as a single uneven object (see Fig. 6.36a).

Accuracy

The accuracy, as expected is at best[3] limited to single pixel resolution for centroid position and object radius.

An interesting observation has been a small *random* deviation (± 2 pixels) in object centroid location, resulting in a similar deviation in object size. At first glance this fluctuation was passed off as an error, however on closer examination it has shown to be able to provide sub-pixel accuracy (through successive averaging) having a pseudo-dithering effect. This can be explained due to an edge effect caused by an imperfectly focused image or a graded object boundary. Subsequently, the static (spatial) fixed-pattern noise (FPN) coupled with

[3]Best performance is achieved in images with high contrast ratio (i.e. dynamic range) and relatively high incident light intensity.

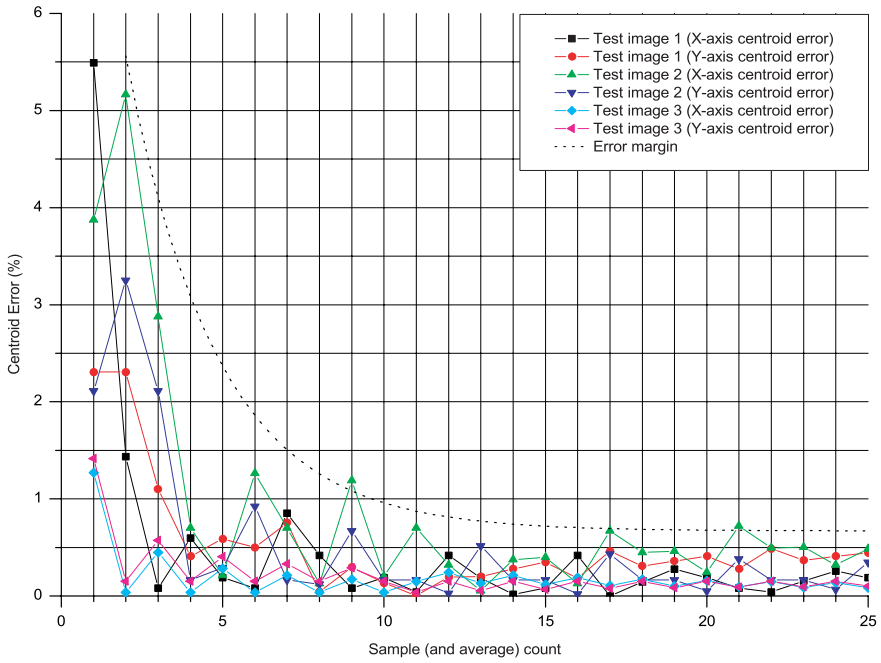


Figure 6.38: Pseudo-dithering providing increased centroid position accuracy through successive averaging.

the (temporal) flicker noise within the edge detector blocks provide this statistically-biased dithering effect. As a result, this provides a mechanism to enable centroid processing time to be tradable with centroid position accuracy, illustrated through trends on measured data shown in Figs. 6.38 and 6.39.

This suggests that less than 1% error is achievable for centroid position and radius measurement by using 10-12 events per result. On this 48×48 pixel array, this corresponds to approximately half pixel accuracy.

6.8.3 Power Consumption

The measured power consumption levels are generally in line with the previously presented simulated results. The measured results partition the total power consumption into the following sources:

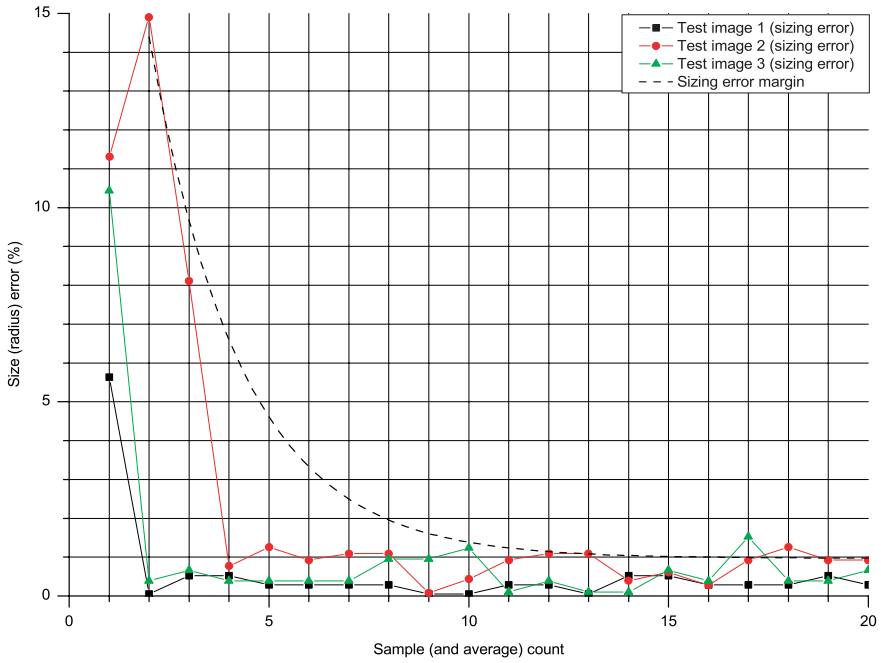


Figure 6.39: Pseudo-dithering providing increased object size accuracy through successive averaging.

Analogue Consumption

This represents the analogue power consumption within the distributed array, including the photocurrents, local and global averaging and threshold/edge detection circuitry. Measured ASP consumption is within 5% of the simulated results. This can be attributed to the fact that all the ASP circuits are biased using current mode techniques, i.e. all inputs are currents. As expected, the ASP power is largely dependant on bias currents and incident light intensity, typically being in the range: 15-50μW, as illustrated in Figs. 6.40 and 6.41.

Digital Consumption (Leakage)

This represents the subthreshold leakage current within the digital core distributed throughout the array. This has been insufficiently considered at design time and has shown to be a main source of power consumption within the ABP core. Power consumption due to this leakage is in the order of 40-60μW.

For such applications with relaxed bandwidth requirements, static leakage can be massively reduced by either increasing the device channel length moderately, or by applying

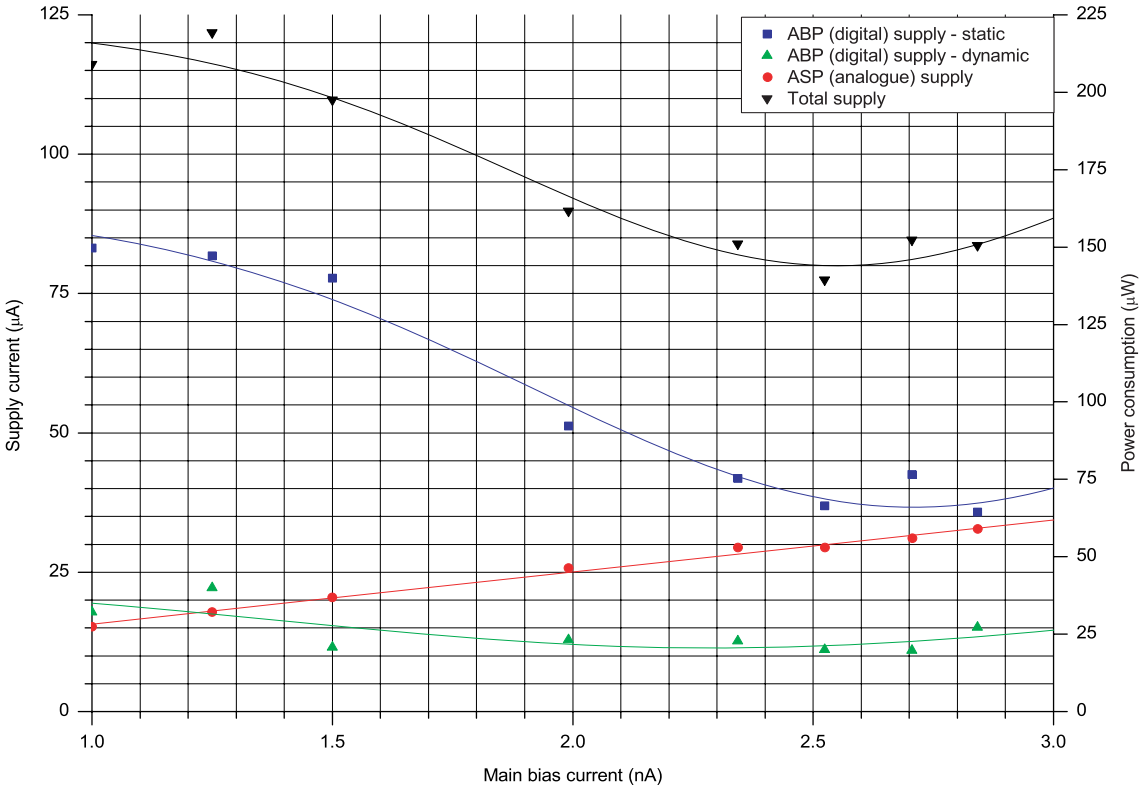


Figure 6.40: Measured supply current levels illustrating the effect of tuning main bias current (feeding edge detectors and discrete delays) on system power consumption.

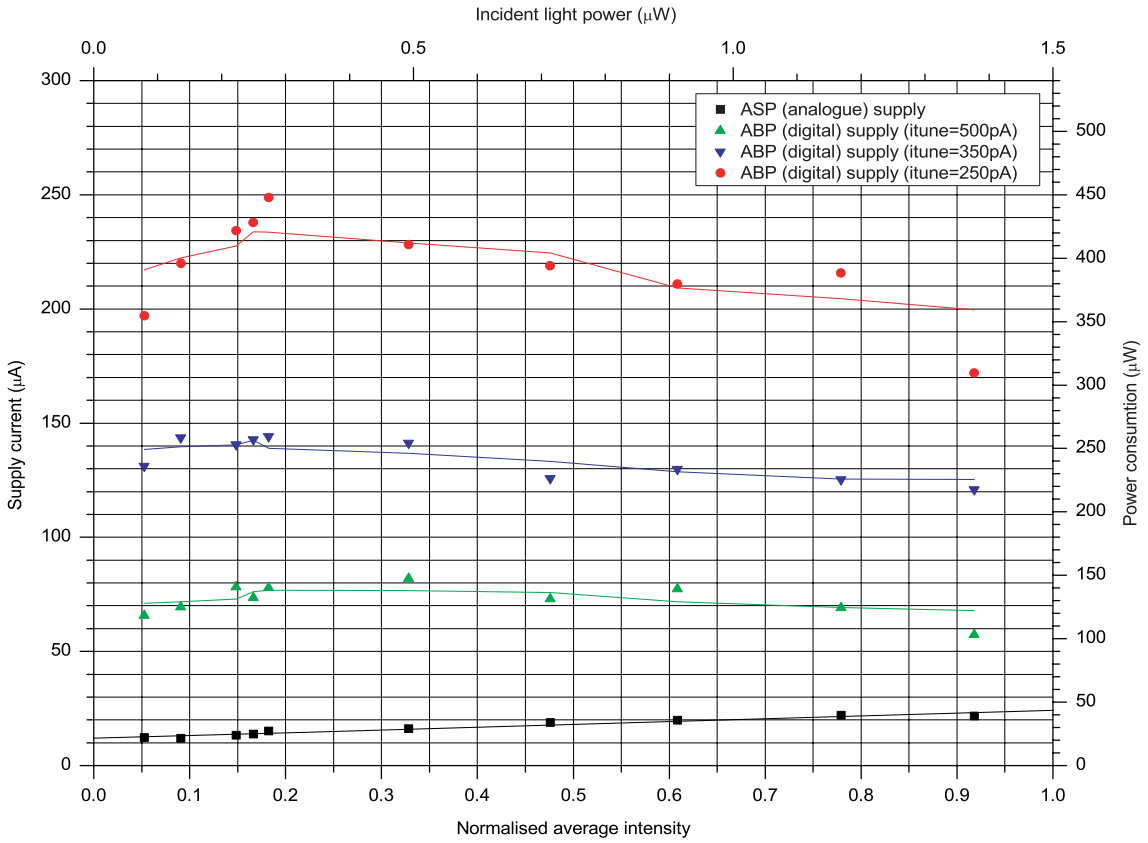


Figure 6.41: Measured supply current levels illustrating the effect of illumination level on system power consumption for various tuning bias current levels (controlling the edge detecting threshold).

a reverse bias on the bulk/source junction. This effect is illustrated in Fig. 6.42. It can therefore be deduced that using NMOS devices of channel length 400nm and PMOS devices of channel length 250nm can reduce static leakage tenfold in comparison to using minimum feature length (180nm) devices.

Digital Consumption (Static)

This represents the static current supply to the digital core distributed throughout the array. This is virtually all due to "digital" short-circuit current caused by incomplete thresholding, i.e. logic gates with non-perfectly discrete inputs. The exact amount of static dissipation is dependant on the configuration of the edge detection circuitry, i.e. bias current levels and input light intensity. This dependance is clearly illustrated in Figs. 6.40 and 6.41.

ABP static consumption could be expected to account for up to 80% of the total system power requirements in certain configurations. However, a significantly lower level of static dissipation has been measured from the expected (simulated) results. The reason for this is that the fixed-pattern noise provides a random offset to the edge detector inputs. Consequently, this inherently biases the differential edge detector output to always have an offset. As the simulations have considered only images of uniform background intensity, this would represent the maximum static dissipation, i.e. when both inputs to a logic gate are not discrete and floating.

Digital Consumption (Dynamic)

This represents power consumption directly related to the distributed binary signal propagation and therefore proportional to the activity. In addition the address-event bus activity influences this level. For typical activities, this has been measured to represent only a 10-15% portion of the total system power consumption. Therefore, no substantial power saving can be achieved by operating the device at a reduced duty cycle.

Other

Power consumption of the I/O cells (obtained from a standard cell library [9]) have not been included, as these have been characterised by the vendor and this consumption is largely dependant on the external circuitry interfacing to the chip, i.e. input capacitances.

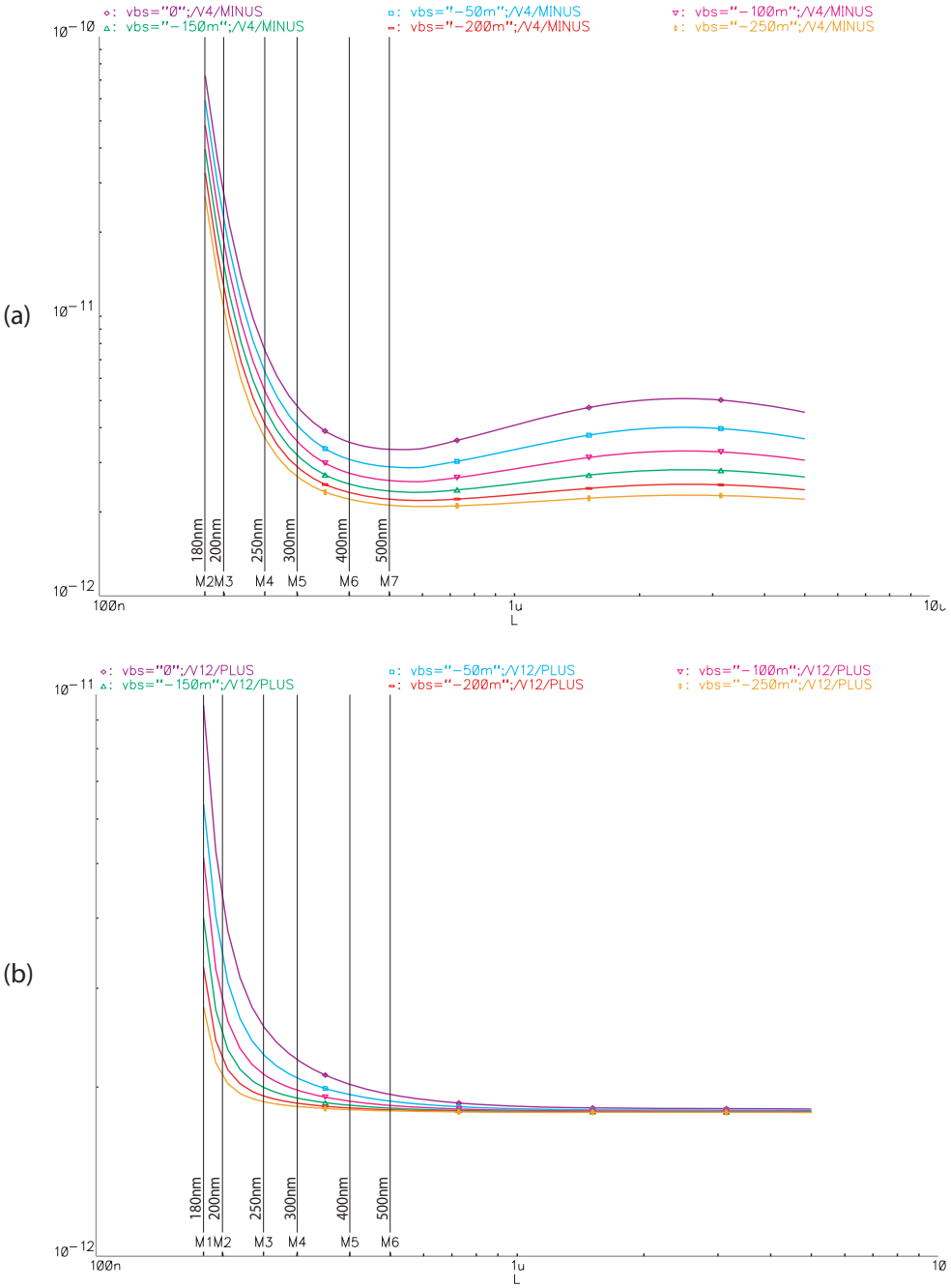


Figure 6.42: Effect of channel length and bulk (reverse) bias on static leakage (off) current (at $V_{ds} = 1.8\text{V}$, $V_{gs} = 0\text{V}$) for (a) NMOS and (b) PMOS devices.

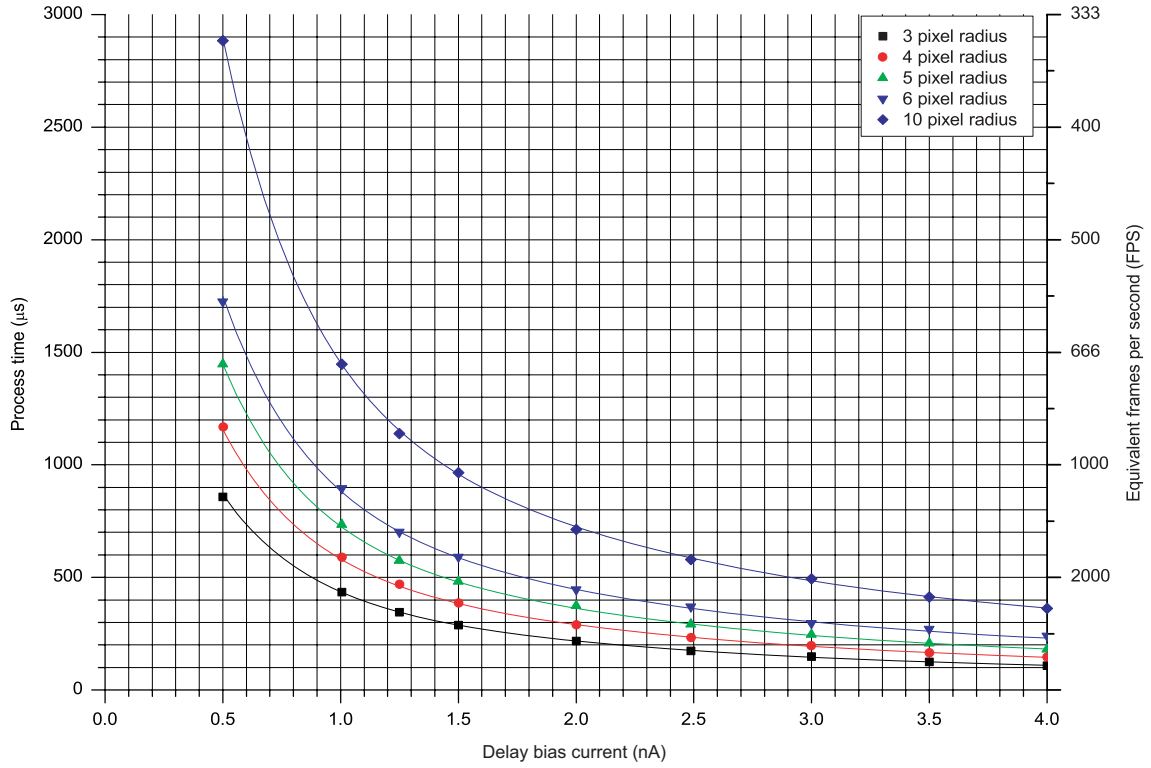


Figure 6.43: Dependance of process time on bias current, given for input images including objects of maximum size of 3, 4, 5, 6 and 8 pixel radius.

6.8.4 Processing Time

Although the asynchronous nature of this distributed system produces temporally unsynchronised events between different objects (due to the local resetting), the algorithm can be run in a "single-shot" mode, and a clock applied to the global reset input. Using this technique a true high frame rate processor can be realised, the limiting factor being the maximum size of detectable object, i.e. the maximum propagation delay. This can in fact be tuned as the internal propagation delay is controlled by a bias current. This relationship between bias current and process time/frame rate is illustrated in Fig. 6.43.

6.8.5 AER Bandwidth

As the Address-Event bus bias is tuneable, the bandwidth dependance on bias current has been measured, illustrated in Fig. 6.44. Since the bus bandwidth requirement for this application is relatively moderate, high bandwidth and channel utilisation is generally not

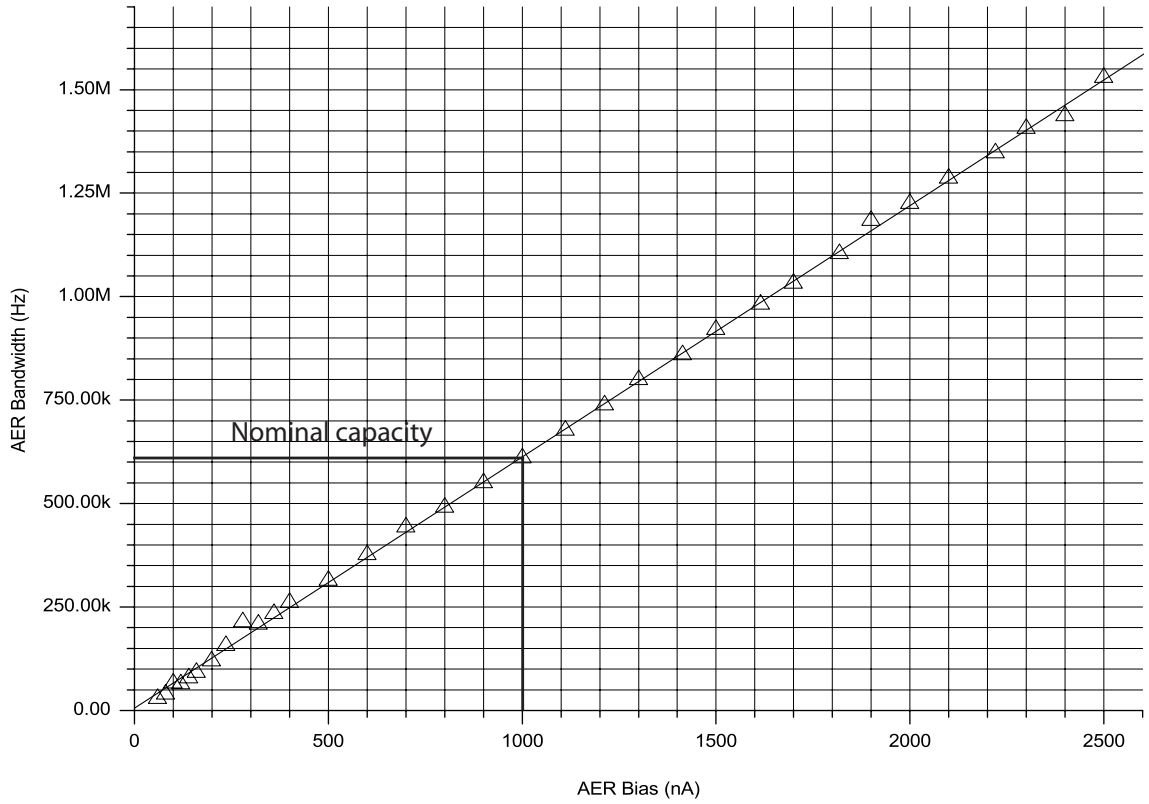


Figure 6.44: Measured address-event bus capacity (bandwidth) by varying pull-up/pull-down bias currents.

an issue. However, for colliding events, the arbitration scheme introduces some latency, and if comparable to the internal propagation delay, this could somewhat distort the information extracted. Therefore a high address-event bandwidth is favourable to achieve a low latency for maintaining temporal resolution.

6.9 Summary

In this chapter a vision processing chip has been presented for object size and centre detection. It is the first system reporting multiple (unlimited) object centroid processing capability. Furthermore the developed system demonstrates high computational efficiency; implementing a computationally intensive algorithm with micropower consumption. Although the developed system includes only a 48×48 cell array; with a cellular power budget of a few tens of nanowatts, scaled to a megapixel array this would only require a few tens of

milliwatts. Also, the fabricated system has shown to utilise fixed pattern noise favourably (as in neurobiology), both reducing power consumption and increasing accuracy through successive sampling. The achieved system specification is summarised in Table. 6.4.

At a component level, novel contributions include a discrete edge detector topology, a locally/globally-averaging threshold detector network and an asynchronous spatiotemporal bio-pulsating core. At an architectural level the contribution is a dedicated vision processor capable of delivering thousands of processed centroids per object every second at ultra-low power levels; at present, unachievable by conventional means.

Technology	UMC 0.18μm MM/RF 1P6M CMOS
Supply voltage	1.8V core (3.3V I/O)
Bias current range	50nA to 2μA (for I_{aer}) 250pA to 10nA (for I_{bias}) 50pA to 2nA (for I_{tune})
Photosensitivity	6 decades, from 100nW/cm^2 to 100mW/cm^2
Responsivity	0.18A/Wcm^2 (for blue light @ $\lambda = 480nm$) 0.28A/Wcm^2 (for green light @ $\lambda = 550nm$) 0.32A/Wcm^2 (for red light @ $\lambda = 650nm$)
Pixel Level	
Pixel size	85$\mu m \times 85\mu m$
Surface fill factor	12.46%
Pixel device count	277
Pixel power	23.04 nW (ASP) 73.44 nW (ABP) 96.48 nW (total)
System Level	
Die dimensions	5mm × 5mm
Array size	48 × 48 pixels
System device count	745,200
System power	222.28 μW (array) 20.3 μW (other) 243.6 μW (total)
Accuracy (centroid and radius)	±1 pixel[1]
Equivalent image process time	0.5ms (maximum)[2]
Address-event bandwidth	0.61 MEPS[3](at I_{aer}=1μA)
Equivalent computational efficiency	1.38 μW per MIPS[2]

[1] Using successive sampling the accuracy can be increased to ±0.5 pixel.
[2] For a test image (with average incident power density of 6$\mu W/cm^2$) consisting of 5 objects of 10 pixel diameter (at I_{bias}=2.5nA, I_{tune}=250pA) .
[3] MEPS = Million Events Per Second

Table 6.4: ORASIS-P2 system properties and performance summary

References

[1] T. G. Constandinou, J. Georgiou and C. Toumazou, "Towards a Bio-inspired Mixed-signal Retinal Processor," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 493–496, 2004.

[2] C. A. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley, 1989.

[3] M. Mahowald, *VLSI Analogs of Neuronal Visual Processing: A Sythesis of Form and Function*. PhD thesis, California Institute of Technology, Pasadena, California, 1992.

[4] C. Toumazou, F. J. Lidgey and D. G. Haigh, *Analog IC Design: The Current-Mode Approach*. London: Peter Perigrinus, 1990.

[5] T. G. Constandinou, J. Georgiou and C. Toumazou, "A Nanopower Tuneable Edge Detection Circuit," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 449–452, 2004.

[6] T. G. Constandinou, J. Georgiou and C. Toumazou, "Nano-power mixed-signal tunable edge-detection circuit for pixel-level processing in next generation vision systems," *IEE Electronics Letters*, vol. 39, no. 25, pp. 1774–1775, 2004.

[7] J. Georgiou, *Micropower Electronics for Neural Prosthetics*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, 2002.

[8] P. Häfliger, *A Spike-based Learning Rule and its Implementation in Analog Hardware"*. PhD thesis, ETH Zürich, Switzerland, 2000.

[9] "L180 60um in-line i/o library," *UMCL18U350T2, Virtual Silicon Corp*, 2002.

Chapter 7

Conclusion

This thesis explores and develops biologically-inspired vision processing using distributed hybrid electronics in CMOS technology.

Chapter 2 introduces neurobiology through system organisation and neural primitives to data representation and spike coding, in particular in reference to the vision system. The notions of biologically-inspired representation and hybrid computation have then been examined in the context of microelectronic integration. Related design and implementation issues for bio-inspired computation have then been outlined specifically in reference to weak inversion analogue and asynchronous binary (or spike domain) computation.

Chapter 3 reviews current state-of-the-art silicon-based imaging technologies and discusses their suitability for integration with processing hardware. This leads to a direct comparison of sequential and distributed topologies for vision processing in custom hardware. A specific vision processing function, centroid detection (and object segmentation) has then been targeted and a comprehensive review of research and development in that arena is given.

7.1 Contributions

Chapter 4 presents a novel distributed algorithm for parallel centroid detection with inherent object segmentation and sizing functionality. It describes the functionality both qualitatively and analytically and provides both experimental verification and an intuitive reasoning behind the high robustness and inherent tolerance to ill-conditioned data and

process variations. A pixel architecture for hardware implementation is proposed and an equivalent software algorithm is coded. Subsequently, the computational load is estimated and power consumption figures are reported by considering benchmark computational efficiencies for state-of-the-art processing hardware. Finally, a generic distributed processing paradigm for hardware implementation is outlined based on the underlying principles of the presented algorithm. The versatility of this array processing platform is reinforced by outlining two specific distributed algorithms directly implementable using this technique.

Chapter 5 reviews silicon-based photodiode modelling specifically related to CMOS pn-junction devices. Furthermore, discussed implementation issues and design techniques for deep submicron technologies are consolidated with measured data from fabricated devices. A review of common photodiode interface topologies is then followed by a novel front-end spiking photoreceptor circuit, with adaptive selection of ON/OFF-encoded channels. The topology is intended for use in adaptable foveating vision chips, where spatial and temporal resolution can be dynamically reconfigured locally.

Chapter 6 describes a novel vision chip implementing the bio-pulsating contour reduction algorithm described previously in chapter 4. This device is the first *silicon retina* reported capable of parallel centroiding of unlimited objects and returning object size in addition to centroid position. The presented system implements a retinocortically-inspired organisation employing an asynchronous binary algorithm combined with continuous time feature extraction. The developed architecture, organisation and circuit topologies are described in detail including both simulated and measured results to validate the theory. The presented system advances vision chip development into exploring new distributed architectures based on hybrid pixels, whilst maintaining micropower consumption and good system stability.

7.2 Recommendations for Future Work

Future developments based on material described in this thesis are proposed in the following areas:

7.2.1 System Optimisation, Enhancement and Development

Although the system has been designed with low power consumption in mind, it could yet further be optimised, also in accuracy, speed and silicon area. At a circuit level, the

current comparators could be redesigned for lower sensitivity to process variation, the global averaging to include both row and column aggregates and the edge detector to be tunable through a single bias. Furthermore, the asynchronous logic can be reduced through custom device-level logic minimisation [1] and static dissipation be reduced by increasing device length moderately.

At an functional level, the edge and contour detection could be improved by implementing a thresholding difference of Gaussian function. Alternatively an adaptive photoreceptor topology [2] could be used to dynamically bias the edge detector block to provide a local automatic gain control and thus improve SNR. Furthermore, the threshold detection could be massively improved by optimising the averaging/smoothing functions.

At a system level, the algorithm could be modified to return the aspect ratio of detectable objects, i.e the W/L. Other techniques for object segmentation could be used to provide versatility to a larger range of object types/input images. For example, colour segmentation [3] could provide a condition for object segmentation where intensity alone fails.

At an technological level, this system is ideally implementable in one of the upcoming 3D CMOS technologies [4] [5], slicing the distributed architecture to several layers, thus increasing fill factor whilst massively reducing both the footprint and interconnectivity requirements. A compact cellular footprint could then open the door to a megapixel resolution vision processor.

7.2.2 Hybrid Distributed Algorithm Design and Implementation

There is great scope to continue work on hybrid distributed algorithms as initiated in this thesis. By dissociating the front-end feature extraction from the higher level back-end algorithm in a distributed fashion as described paves the way for implementing countless computationally demanding algorithms. Moreover, towards the end of chapter 4, two specific examples including such hardware implementable algorithms have been proposed.

Ultimately, the proposed hybrid distributed architecture could be extended to implement an *FPGA-like* vision processor with a reconfigurable *back-end* for providing a generic platform for custom binary algorithm implementation. This would provide the perfect compliment to the *front-end* reconfigurability already developed within the Cellular Neural Network (CNN) community [6] [7] [8].

References

[1] N. H. E. Weste and K. Eshraghian, *Priciples of CMOS VLSI design: A systems perspective*. Addison-Wesley, 1993.

[2] T. Delbrck and C. A. Mead, "Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits," *Vision Chips: Implementing vision algorithms with analog VLSI circuits, by C. Koch and H. Li eds.*, pp. 139–161, 1995.

[3] R. Merrill, "Color separation in an active pixel cell imaging array using a triple-well structure." US Patent Number 5,965,875, 1999.

[4] R. Islam, C. Brubaker, P. Lindner and C. Schaefer, "Wafer level packaging and 3D interconnect for IC technology," *IEEE/SEMI Conference and Workshop on Advanced Semiconductor Manufacturing*, pp. 212–217, 2002.

[5] J. Baliga, "Chips go vertical [3D IC interconnection]," *IEEE Spectrum*, vol. 41, no. 3, pp. 43–47, 2004.

[6] T. Roska and A. Rodríguez-Vázquez, eds., *Towards the Visual Microprocessor: VLSI Design and the Use of Cellular Neural Network Universal Machines*. Wiley, 2001.

[7] G. L Cembrano, A. Rodriguez-Vazquez, R. C. Galan, F. Jimenez-Garrido, S. Espejo and R Dominguez-Castro, "A 1000 FPS at 128x128 Vision Processor With 8-Bit Digitized I/O," *IEEE Journal of Solid State Circuits*, vol. 39, no. 2, pp. 1044–1055, 2004.

[8] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro and S. E. Meana, "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Towards VSoCs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 851–863, 2004.

Appendix A

Algorithm Simulation Source Code

```
{*************************************************************************
 Program : ORASIS Simulator
  Module : MAIN.PAS
    Date : See File Timestamp
  Author : Timothy G Constandinou
 Company : Imperial College London
 *************************************************************************}

unit main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Buttons, ExtDlgs, Menus, ComCtrls, Grids, Math;

const
  xgrid            = 250;
  ygrid            = 250;
  maxresets        = 100000;
  maxsize          = 27;
  defaultfpn       = 10;
  defaultdefects   = 10;
  defaultedge      = 128;
  defaultthreshold = 128;
  defaultbgcolour  = 0;

type
  Tmainform = class(TForm)
    Image1: TImage; Image2: TImage; Image3: TImage; Image4: TImage;
    Image5: TImage; Image6: TImage; Image7: TImage; Label1: TLabel;
    Label2: TLabel; Label3: TLabel; Label4: TLabel; Label5: TLabel;
    Label6: TLabel; Label8: TLabel; OpenPictureDialog1: TOpenPictureDialog;
    MainMenu1: TMainMenu; FreqGrid: TStringGrid; N1: TMenuItem; N2: TMenuItem;
    N3: TMenuItem; File1: TMenuItem; Open1: TMenuItem; Exit1: TMenuItem;
    Mode: TMenuItem; Scan1: TMenuItem; Help1: TMenuItem; About1: TMenuItem;
    N01fps: TMenuItem; N1fps: TMenuItem; N5fps: TMenuItem; N10fps: TMenuItem;
    MaxRefresh1: TMenuItem; SaveImages1: TMenuItem; Monochrome1: TMenuItem;
    SingleFrame1: TMenuItem; Random1: TMenuItem; NoiseBox1: TGroupBox;
    GroupBox1: TGroupBox; ThresholdsBox: TGroupBox; AddFlatButton: TButton;
    AddGaussianButton: TButton; AddSpeckleButton: TButton; Label10: TLabel;
    Label11: TLabel; FPN: TEdit; Defects: TEdit; AddSaltButton: TButton;
    AddPepperButton: TButton; AddSaltPepperButton: TButton; Label7: TLabel;
    ResetAllButton: TButton; AutoThresholdButton: TButton; Label9: TLabel;
    SetThresholdsButton: TButton; GlobalThresholdLabel: TLabel;
    EdgeThresholdLabel: TLabel; GlobalThreshold: TTrackBar; Label12: TLabel;
    EdgeThreshold: TTrackBar; ResultDesc: TEdit; AppendFileButton: TButton;

    procedure FormCreate(Sender: TObject);
```

```
    procedure IdleHandler(Sender: TObject; var Done: Boolean);
    procedure Exit1Click(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure GlobalThresholdChange(Sender: TObject);
    procedure EdgeThresholdChange(Sender: TObject);
    procedure AutoThresholdButtonClick(Sender: TObject);
    procedure AddFlatButtonClick(Sender: TObject);
    procedure ResetAllButtonClick(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure AddSaltButtonClick(Sender: TObject);
    procedure Scan1Click(Sender: TObject);
    procedure Random1Click(Sender: TObject);
    procedure SetThresholdsButtonClick(Sender: TObject);
    procedure MaxRefresh1Click(Sender: TObject);
    procedure N10fpsClick(Sender: TObject);
    procedure N5fpsClick(Sender: TObject);
    procedure N1fpsClick(Sender: TObject);
    procedure N01fpsClick(Sender: TObject);
    procedure SingleFrame1Click(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure SaveImages1Click(Sender: TObject);
    procedure Monochrome1Click(Sender: TObject);
    procedure AddPepperButtonClick(Sender: TObject);
    procedure AddSaltPepperButtonClick(Sender: TObject);
    procedure AddSpeckleButtonClick(Sender: TObject);
    procedure AddGaussianButtonClick(Sender: TObject);
    procedure AppendFileButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

  TPixelElement = record Pixel : integer; Status : boolean; end;
  TSizeElement = record Count, Size : integer; end;
  TPixelArray = array[1..xgrid, 1..ygrid] of TPixelElement;
  TSizeArray = array[1..xgrid, 1..ygrid] of TSizeElement;
  TReset = record x, y : array[1..maxresets] of integer; n : integer; end;

var
  mainform      : TMainForm;
  OutFile       : textfile;
  SizeArray     : TSizeArray;
  Centre, Reset : TReset;
  DoResetAll    : Boolean;
  CurrGen       : integer;
  OrigArray, PixelArray, PixelArray2 : TPixelArray;
  BgCol, FgCol1, FgCol2, FgCol3, FgCol4, FgCol5: TColor;
```

```
implementation

uses about; {$R *.DFM}

{ GENERIC FUNCTIONS }

  procedure Mul(var a : integer; b : double);
    begin a := round(a * b); end;

  procedure Delay(msecs:integer);
    var
      FirstTickCount:longint;
    begin
      FirstTickCount:=GetTickCount;
      repeat
        Application.ProcessMessages; {allowing access to other controls, etc.}
      until ((GetTickCount-FirstTickCount) >= longint(msecs));
    end;

  function Convert24bitTo8bitGrey(incolor : integer) : integer;
    begin
      result := 0;
      while (incolor > 65536) do
        begin incolor := incolor - 65536; inc(Result); end;
      while (incolor > 256) do
        begin incolor := incolor - 256; inc(Result); end;
      Result := (Result + incolor) div 3;
    end;

  function Convert8bitGreyTo24bit(incolor : integer) : integer;
    begin result := (incolor * 65536) + (incolor * 256) + incolor; end;

  procedure EmptySizeArray(var TempArray : TSizeArray);
    var
      x, y : integer;
    begin
      for y := 1 to XGrid do
        for x := 1 to YGrid do
          begin
            TempArray[x,y].count := 0;
            TempArray[x,y].size  := 0;
          end;
    end;

  procedure EmptyPixelArray(var TempArray : TPixelArray);
    var
      x, y : integer;
    begin
      for y := 1 to XGrid do
```

```
        for x := 1 to YGrid do
          begin
            TempArray[x,y].Pixel  := 0;
            TempArray[x,y].Status := FALSE;
          end;
    end;

  function GetFPS() : integer;
    begin
      Result := 0;
      if mainform.N10fps.checked then Result := 100;
      if mainform.N5fps.checked then  Result := 200;
      if mainform.N1fps.checked then  Result := 1000;
      if mainform.N01fps.checked then Result := 10000;
    end;

{ PROCESSING PROCEDURES }

  procedure ImageToArray();
    var
      x, y : integer;
    begin
      for y := 1 to mainform.Image1.height do
        for x := 1 to mainform.Image1.width do
          PixelArray[x*2-1,y*2-1].Pixel :=
          Convert24bitTo8bitGrey(mainform.Image1.Canvas.Pixels[x,y]);
    end;

  function isEdge(x1, y1, x2, y2 : integer) : boolean;
    begin
      Result := FALSE;
      if (abs(PixelArray[x1, y1].Pixel - PixelArray[x2, y2].Pixel) >
      mainform.EdgeThreshold.Position) then Result := TRUE;
    end;

  procedure CalcEdges();
    var
      x, y : integer;
    begin
      for y := 1 to mainform.Image1.height do
        for x := 1 to mainform.Image1.width do
          begin
            PixelArray[x*2, y*2-1].Status := IsEdge(x*2-1, y*2-1, x*2+1, y*2-1);
            PixelArray[x*2-1, y*2].Status := IsEdge(x*2-1, y*2-1, x*2-1, y*2+1);
          end;
    end;

  procedure CalcEdgeNodes(var TempPixelArray:TPixelArray);
    var
```

```
    x, y, temp : integer;
  begin
    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        begin
          temp := 0;
          if TempPixelArray[x*2-1, y*2].Status then inc(Temp);
          if TempPixelArray[x*2+1, y*2].Status then inc(Temp);
          if TempPixelArray[x*2, y*2-1].Status then inc(Temp);
          if TempPixelArray[x*2, y*2+1].Status then inc(Temp);
          if (temp = 2) and (PixelArray[x*2, y*2].Pixel >
                              mainform.GlobalThreshold.Position) then
            TempPixelArray[x*2, y*2].Status := TRUE;
        end;
  end;

procedure CalcAverages();
  var
    x, y, i, j, av : integer;
  begin
    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        begin
          av := 0;
          for j := 1 to 4 do for i := 1 to 4 do
              inc(av, PixelArray[x*2+(i*2)-5, y*2+(j*2)-5].Pixel);
          PixelArray[x*2, y*2].Pixel := av div 16;
        end;
  end;

procedure CalcGlobalAverage();
  var
    x, y, temp : integer;
  begin
    temp := 0;

    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        temp := temp + PixelArray[x*2-1, y*2-1].Pixel;

    temp := temp div (mainform.Image1.height * mainform.Image1.width);
    mainform.GlobalThreshold.Position := temp;
  end;

procedure ResetPixel(x, y : integer);
  begin
   inc(reset.n);
   reset.x[reset.n] := x;
   reset.y[reset.n] := y;
```

```
    PixelArray[x,y].Status := FALSE;
    PixelArray2[x,y].Status := FALSE;
    if (x>2) and PixelArray2[x-2,y].Status then ResetPixel(x-2,y);
    if (y>2) and PixelArray2[x,y-2].Status then ResetPixel(x,y-2);
    if (x+2<(2*mainform.image1.width)) and PixelArray2[x+2,y].Status then
      ResetPixel(x+2,y);
    if (y+2<(2*mainform.image1.height)) and PixelArray2[x,y+2].Status then
      ResetPixel(x,y+2);
  end;

procedure CalcStaticNoise(NoiseType : string);
  var
    x, y, randlimit : integer;
    randspread, temp : double;
  begin
    randomize;
    randlimit := round(2.55 * strtofloat(mainform.FPN.text));
    randspread := (strtofloat(mainform.FPN.text) / 100);

    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        begin
          if (NoiseType='flat') then
            begin
              temp := randlimit / 2 - random(randlimit);
              inc(PixelArray[x*2-1,y*2-1].Pixel, round(temp));
            end;
          if (NoiseType='gaussian') then
            begin
              temp := RandG(0, (randlimit / 6));
              inc(PixelArray[x*2-1,y*2-1].Pixel, round(temp));
            end;
          if (NoiseType='speckle') then
            begin
              temp := 1 + randspread * (random - 0.5);
              mul(PixelArray[x*2-1,y*2-1].Pixel, temp);
            end;
          if (PixelArray[x*2-1,y*2-1].Pixel > 255) then
            PixelArray[x*2-1,y*2-1].Pixel := 255;
          if (PixelArray[x*2-1,y*2-1].Pixel < 0) then
            PixelArray[x*2-1,y*2-1].Pixel := 0;
        end;
    Mainform.ResultDesc.Text := Mainform.ResultDesc.Text + NoiseType + '=' +
                                mainform.FPN.text + ', ';
  end;

procedure CalcSaltPepperNoise(NoiseType : string);
  var
```

```
    n, x, y, defects : integer;
  begin
    randomize;
    defects := round(strtofloat(mainform.defects.text));

    for n := 1 to defects do
      begin
        x := random(mainform.Image1.width);
        y := random(mainform.Image1.height);
        if (NoiseType='salt') then PixelArray[x*2-1,y*2-1].Pixel := 255;
        if (NoiseType='pepper') then PixelArray[x*2-1,y*2-1].Pixel := 0;
      end;
    Mainform.ResultDesc.Text := Mainform.ResultDesc.Text + NoiseType + '=' +
                                inttostr(defects) + ', ';
  end;

procedure CalcNextGen;
  var
    x, y : integer;
  begin
    centre.n := 0; reset.n  := 0; PixelArray2 := PixelArray;

    for y := 2 to mainform.Image1.height-1 do
      for x := 2 to mainform.Image1.width-1 do
        begin
          if (PixelArray[x*2, y*2].Pixel < mainform.GlobalThreshold.Position)
          and not(PixelArray[x*2, y*2].Status) then
            begin
              if PixelArray[(x-1)*2, y*2].Status or
              PixelArray[(x+1)*2, y*2].Status or
              PixelArray[x*2, (y-1)*2].Status or
              PixelArray[x*2, (y+1)*2].Status then
                PixelArray2[x*2, y*2].Status := TRUE;

              if (PixelArray[(x-3)*2, y*2].Status and
              PixelArray[(x+3)*2, y*2].Status) and
              (PixelArray[x*2, (y-3)*2].Status and
              PixelArray[x*2, (y+3)*2].Status) then
                begin
                  centre.n := centre.n + 1;
                  centre.x[centre.n] := x * 2;
                  centre.y[centre.n] := y * 2;
                  ResetPixel(x*2, y*2);
                  CalcEdgeNodes(PixelArray2);
                end;
            end;

        end;
    PixelArray := PixelArray2;
```

```
    end;

  procedure CalcNextGenRandom;
    var
      x, y, count, maxcount : integer;
      TempArray             : TPixelArray;
    begin
      centre.n := 0; reset.n := 0; count := 0; PixelArray2 := PixelArray;
      maxcount := (mainform.Image1.width - 2) * (mainform.Image1.height - 2);
      EmptyPixelArray(TempArray);

      while (count < maxcount) do
        begin
          x := 1 + random(mainform.Image1.width - 1);
          y := 1 + random(mainform.Image1.height - 1);
          if not(TempArray[x,y].Status) then
            begin
              TempArray[x,y].Status := TRUE;
              inc(count);

              if (PixelArray[x*2, y*2].Pixel < mainform.GlobalThreshold.Position)
              and not(PixelArray[x*2, y*2].Status) then
                begin
                  if PixelArray[(x-1)*2, y*2].Status or
                  PixelArray[(x+1)*2, y*2].Status or
                  PixelArray[x*2, (y-1)*2].Status or
                  PixelArray[x*2, (y+1)*2].Status then
                    PixelArray2[x*2, y*2].Status := TRUE;

                  if (PixelArray[(x-3)*2, y*2].Status and
                  PixelArray[(x+3)*2, y*2].Status) and
                  (PixelArray[x*2, (y-3)*2].Status and
                  PixelArray[x*2, (y+3)*2].Status) then
                    begin
                      inc(centre.n);
                      centre.x[centre.n] := x * 2;
                      centre.y[centre.n] := y * 2;
                      ResetPixel(x * 2, y * 2);
                      CalcEdgeNodes(PixelArray2);
                    end;
                end;
            end;
        end;
      PixelArray := PixelArray2;
    end;

{ DISPLAY PROCEDURES }

  procedure DisplayImage();
```

```
  var
    x, y : integer;
  begin
    mainform.Image1.Canvas.Brush.Color := BgCol;
    mainform.Image1.Canvas.FillRect
    (Rect(0,0,mainform.Image1.Width, mainform.Image1.Height));

    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        mainform.Image1.Canvas.Pixels[x,y] :=
        Convert8bitGreyTo24bit(PixelArray[x*2-1, y*2-1].Pixel);
  end;

procedure DisplayAverages();
  var
    x, y : integer;
  begin
    mainform.Image2.Canvas.Brush.Color := BgCol;
    mainform.Image2.Canvas.FillRect
    (Rect(0,0,mainform.Image2.Width, mainform.Image2.Height));

    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        mainform.Image2.Canvas.Pixels[x,y] :=
        Convert8bitGreyTo24bit(PixelArray[x*2, y*2].Pixel);
  end;

procedure DisplayThreshold();
  var
    x, y : integer;
  begin
    mainform.Image3.Canvas.Brush.Color := BgCol;
    mainform.Image3.Canvas.FillRect
    (Rect(0,0,mainform.Image3.Width, mainform.Image3.Height));

    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        if (PixelArray[x*2, y*2].Pixel<mainform.GlobalThreshold.Position) then
          mainform.Image3.Canvas.Pixels[x,y] := FgCol1;
  end;

procedure DisplayEdges();
  var
    x, y : integer;
  begin
    mainform.Image4.Canvas.Brush.Color := BgCol;
    mainform.Image4.Canvas.FillRect
    (Rect(0,0,mainform.Image4.Width, mainform.Image4.Height));
```

```
    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        if PixelArray[x*2, y*2].Status then
          mainform.Image4.Canvas.Pixels[x,y] := FgCol2
  end;

procedure DisplayNextGen();
  var
    x, y : integer;
  begin
    mainform.Image5.Canvas.Brush.Color := BgCol;
    mainform.Image5.Canvas.FillRect
    (Rect(0,0,mainform.Image5.Width, mainform.Image5.Height));

    for y := 1 to mainform.Image1.height do
      for x := 1 to mainform.Image1.width do
        if PixelArray[x*2, y*2].Status then
          mainform.Image5.Canvas.Pixels[x,y] := FgCol3;

    if mainform.Saveimages1.Checked and (currgen = 1) then
      begin
        mainform.Image2.Picture.SaveToFile('smoothed.bmp');
        mainform.Image3.Picture.SaveToFile('threshold.bmp');
        mainform.Image4.Picture.SaveToFile('contour.bmp');
      end;

    if mainform.Saveimages1.Checked and (currgen < 100) then
      mainform.Image5.Picture.SaveToFile('firing'+inttostr(currgen)+'.bmp');
  end;

procedure DisplayResets();
  var
    n  : integer;
  begin
    mainform.Image6.Canvas.Brush.Color := BgCol;
    mainform.Image6.Canvas.FillRect
    (Rect(0,0,mainform.Image6.Width, mainform.Image6.Height));

    for n := 1 to reset.n do
      mainform.Image6.Canvas.Pixels[(reset.x[n] div 2),
      (reset.y[n] div 2)] := FgCol4;

    for n := 1 to centre.n do
      begin
        mainform.Image6.Canvas.Pixels[(centre.x[n] div 2),
        (centre.y[n] div 2)] := FgCol5;
        mainform.Image6.Canvas.Pixels[(centre.x[n] div 2)-1,
        (centre.y[n] div 2)] := FgCol5;
        mainform.Image6.Canvas.Pixels[(centre.x[n] div 2)+1,
```

```
        (centre.y[n] div 2)] := FgCol5;
        mainform.Image6.Canvas.Pixels[(centre.x[n] div 2),
        (centre.y[n] div 2)-1] := FgCol5;
        mainform.Image6.Canvas.Pixels[(centre.x[n] div 2),
        (centre.y[n] div 2)+1] := FgCol5;
      end;
  end;

procedure DisplayCentres();
  var
    n  : integer;
  begin
    mainform.Image7.Canvas.Brush.Color := BgCol;
    mainform.Image7.Canvas.FillRect
    (Rect(0,0,mainform.Image7.Width, mainform.Image7.Height));

    for n := 1 to Centre.n do
      mainform.Image7.Canvas.Pixels[(centre.x[n] div 2),
      (centre.y[n] div 2)] := FgCol5;
  end;

procedure DisplaySizes();
  var
    x, y  : integer;
    count : array[1..maxsize] of integer;
  begin
    if mainform.Scan1.checked then
      begin
        for y := 1 to Ygrid do
          for x := 1 to XGrid do
            inc(SizeArray[x,y].Count);

        for x := 1 to centre.n do
          begin
            SizeArray[centre.x[x], centre.y[x]].Size :=
            SizeArray[centre.x[x], centre.y[x]].Count;
            SizeArray[centre.x[x], centre.y[x]].Count := 0;
          end;

        for x := 1 to maxsize do count[x] := 0;

        for y := 1 to Ygrid do
          for x := 1 to XGrid do
            if (SizeArray[x, y].Count>maxsize) then SizeArray[x, y].Size := 0
              else
            if (SizeArray[x, y].Size>0) then inc(Count[SizeArray[x, y].Size]);
      end
    else
      for x := 1 to maxsize do count[x] := 0;
```

```
        y := 0;

        for x := 1 to maxsize do
          begin
            y := y + count[x];
            if (mainform.FreqGrid.Cells[x, 1] <> inttostr(count[x])) then
              mainform.FreqGrid.Cells[x, 1] := inttostr(count[x]);
          end;

        If (mainform.FreqGrid.Cells[maxsize + 1, 1] <> inttostr(y)) then
          mainform.FreqGrid.Cells[maxsize + 1, 1] := inttostr(y);
    end;


{ FILE OUTPUT PROCEDURES }

  procedure OutputHeader();
    begin
      write(outfile, DateTimeToStr(Now)+': ');
      write(outfile, 'filename="'+mainform.OpenPictureDialog1.FileName+'", ');
      write(outfile, 'edge='+inttostr(Mainform.EdgeThreshold.Position)+', ');
      write(outfile, 'threshold='+inttostr(Mainform.GlobalThreshold.Position));
      writeln(outfile, ', '+mainform.ResultDesc.Text);
    end;

  procedure OutputResults();
    var
      n : integer;
    begin
      for n := 1 to maxsize do
        write(outfile, mainform.FreqGrid.Cells[n, 1]+' ');
      writeln(outfile, mainform.FreqGrid.Cells[maxsize+1, 1]);
    end;

  procedure OutputFile();
    begin
      assignfile(outfile, 'results.txt');
      if fileexists('results.txt') then append(outfile) else rewrite(outfile);
      OutputHeader(); OutputResults();
      closefile(outfile);
    end;


{ INITIALISATION AND CLEARUP PROCEDURES }

  procedure InitFreqGrid();
    var
      n : integer;
```

```
  begin
    mainform.FreqGrid.Cells[0,0] := 'Size';
    mainform.FreqGrid.Cells[0,1] := 'Count';
    mainform.FreqGrid.Cells[(maxsize+1),0] := 'Total';

    for n := 1 to maxsize do mainform.FreqGrid.Cells[n, 0] := inttostr(n);
  end;

procedure ClearImages();
  begin
    mainform.Image1.Canvas.Brush.Color := BgCol;
    mainform.Image1.Canvas.FillRect
    (Rect(0,0,mainform.Image1.Width, mainform.Image1.Height));
    mainform.Image1.refresh;
    mainform.Image2.Canvas.Brush.Color := BgCol;
    mainform.Image2.Canvas.FillRect
    (Rect(0,0,mainform.Image2.Width, mainform.Image2.Height));
    mainform.Image2.refresh;
    mainform.Image3.Canvas.Brush.Color := BgCol;
    mainform.Image3.Canvas.FillRect
    (Rect(0,0,mainform.Image3.Width, mainform.Image3.Height));
    mainform.Image3.refresh;
    mainform.Image4.Canvas.Brush.Color := BgCol;
    mainform.Image4.Canvas.FillRect
    (Rect(0,0,mainform.Image4.Width, mainform.Image4.Height));
    mainform.Image4.refresh;
    mainform.Image5.Canvas.Brush.Color := BgCol;
    mainform.Image5.Canvas.FillRect
    (Rect(0,0,mainform.Image5.Width, mainform.Image5.Height));
    mainform.Image5.refresh;
    mainform.Image6.Canvas.Brush.Color := BgCol;
    mainform.Image6.Canvas.FillRect
    (Rect(0,0,mainform.Image6.Width, mainform.Image6.Height));
    mainform.Image6.refresh;
    mainform.Image7.Canvas.Brush.Color := BgCol;
    mainform.Image7.Canvas.FillRect
    (Rect(0,0,mainform.Image7.Width, mainform.Image7.Height));
    mainform.Image7.refresh;
  end;

procedure ResetAll(CalcGlobalThreshold : boolean);
  begin
    CurrGen := 0;
    EmptySizeArray(SizeArray); EmptyPixelArray(PixelArray);
    ImageToArray(); CalcAverages();
    if CalcGlobalThreshold then CalcGlobalAverage();
    CalcEdges(); CalcEdgeNodes(PixelArray);

    DisplayImage(); Delay(50); DisplayAverages(); Delay(50);
```

```
      DisplayThreshold(); Delay(50); DisplayEdges(); Delay(50);
      DisplayNextGen(); Delay(50); DisplayResets(); Delay(50);
      DisplayCentres(); Delay(50); DisplaySizes();
    end;

{ TRACKBAR EVENT HANDLERS }

  procedure Tmainform.GlobalThresholdChange(Sender: TObject);
    begin
      if (GlobalThresholdLabel.Caption <> inttostr(GlobalThreshold.Position)) then
        begin
          GlobalThresholdLabel.Caption := inttostr(GlobalThreshold.Position);
          ResetAll(FALSE);
        end;
    end;

  procedure Tmainform.EdgeThresholdChange(Sender: TObject);
    begin
      if (EdgeThresholdLabel.Caption <> inttostr(EdgeThreshold.Position)) then
        begin
          EdgeThresholdLabel.Caption := inttostr(EdgeThreshold.Position);
          ResetAll(FALSE);
        end;
    end;

{ BUTTON EVENT HANDLERS }

  procedure Tmainform.ResetAllButtonClick(Sender: TObject);
    begin
      DoResetAll                        := TRUE;
      ClearImages();
//    mainform.GlobalThreshold.Position := DefaultThreshold;
//    mainform.EdgeThreshold.Position   := DefaultEdge;
      mainform.ResultDesc.Text          := '';
      PixelArray                        := OrigArray;
      DisplayImage; ResetAll(TRUE);
      DoResetAll                        := FALSE;
    end;

  procedure Tmainform.AutoThresholdButtonClick(Sender: TObject);
    begin ResetAll(TRUE); end;

  procedure Tmainform.SetThresholdsButtonClick(Sender: TObject);
    begin ResetAll(FALSE); end;

  procedure Tmainform.AddFlatButtonClick(Sender: TObject);
    begin CalcStaticNoise('flat'); DisplayImage(); ResetAll(TRUE); end;

  procedure Tmainform.AddGaussianButtonClick(Sender: TObject);
```

```
    begin CalcStaticNoise('gaussian'); DisplayImage(); ResetAll(TRUE); end;

  procedure Tmainform.AddSpeckleButtonClick(Sender: TObject);
    begin CalcStaticNoise('speckle'); DisplayImage(); ResetAll(TRUE); end;

  procedure Tmainform.AddSaltButtonClick(Sender: TObject);
    begin CalcSaltPepperNoise('salt'); DisplayImage(); ResetAll(TRUE); end;

  procedure Tmainform.AddPepperButtonClick(Sender: TObject);
    begin CalcSaltPepperNoise('pepper'); DisplayImage(); ResetAll(TRUE); end;

  procedure Tmainform.AddSaltPepperButtonClick(Sender: TObject);
    begin
      CalcSaltPepperNoise('salt'); DisplayImage();
      CalcSaltPepperNoise('pepper'); DisplayImage(); ResetAll(TRUE);
    end;

  procedure Tmainform.AppendFileButtonClick(Sender: TObject);
    begin OutputFile(); end;


{ MENU EVENT HANDLERS }

  procedure Tmainform.Open1Click(Sender: TObject);
    begin
      if OpenPictureDialog1.Execute then
        begin
          Image1.Picture.LoadFromFile(OpenPictureDialog1.Filename);
          ResetAll(TRUE); OrigArray := PixelArray;
         end;
    end;

  procedure Tmainform.Exit1Click(Sender: TObject);
    begin Application.Terminate; end;

  procedure Tmainform.MaxRefresh1Click(Sender: TObject);
    begin mainform.MaxRefresh1.Checked := TRUE; end;

  procedure Tmainform.N10fpsClick(Sender: TObject);
    begin mainform.N10fps.Checked := TRUE; end;

  procedure Tmainform.N5fpsClick(Sender: TObject);
    begin mainform.N5fps.Checked := TRUE; end;

  procedure Tmainform.N1fpsClick(Sender: TObject);
    begin mainform.N1fps.Checked := TRUE; end;

  procedure Tmainform.N01fpsClick(Sender: TObject);
    begin mainform.N01fps.Checked := TRUE; end;
```

```
  procedure Tmainform.SingleFrame1Click(Sender: TObject);
    begin mainform.SingleFrame1.Checked := TRUE; end;

  procedure Tmainform.Scan1Click(Sender: TObject);
    begin mainform.Scan1.Checked := True; ResetAll(False); end;

  procedure Tmainform.Random1Click(Sender: TObject);
    begin mainform.Random1.Checked := True; ResetAll(False); end;

  procedure Tmainform.About1Click(Sender: TObject);
    begin aboutform.show; end;

  procedure Tmainform.SaveImages1Click(Sender: TObject);
    begin mainform.Saveimages1.Checked:=not(mainform.Saveimages1.Checked); end;

  procedure Tmainform.Monochrome1Click(Sender: TObject);
    begin
      mainform.Monochrome1.Checked:=not(mainform.Monochrome1.Checked);
      if mainform.Monochrome1.Checked then
        begin
          BgCol  := $00FFFFFF; FgCol1 := $00000000; FgCol2 := $00000000;
          FgCol3 := $00000000; FgCol4 := $00999999; FgCol5 := $00000000;
        end
      else
        begin
          BgCol  := $00000000; FgCol1 := $00FF0000; FgCol2 := $0000FF00;
          FgCol3 := $000000FF; FgCol4 := $00666666; FgCol5 := $00FFFFFF;
       end;
      ResetAll(TRUE);
    end;

{ APPLICATION EVENT HANDLERS }

  procedure Tmainform.IdleHandler(Sender: TObject; var Done: Boolean);
    begin
      if not(DoResetAll) and not(mainform.SingleFrame1.Checked) then
        begin
          Delay(GetFPS());
          if mainform.scan1.checked then CalcNextGen() else CalcNextGenRandom();
          DisplayNextGen(); DisplayResets(); DisplayCentres(); DisplaySizes();
          inc(CurrGen);
        end;
    end;

  procedure Tmainform.Image1Click(Sender: TObject);
    begin
      if mainform.SingleFrame1.Checked then
        begin
```

```
          if mainform.scan1.checked then CalcNextGen() else CalcNextGenRandom();
          DisplayNextGen(); DisplayResets(); DisplayCentres(); DisplaySizes();
        end;
    end;

  procedure Tmainform.FormCreate(Sender: TObject);
    begin
      BgCol  := $00000000; FgCol1 := $00FF0000; FgCol2 := $0000FF00;
      FgCol3 := $000000FF; FgCol4 := $00666666; FgCol5 := $00FFFFFF;
      DoResetAll                          := TRUE;
      Application.OnIdle                  := IdleHandler;
      InitFreqGrid();
      mainform.GlobalThreshold.Position := defaultThreshold;
      mainform.EdgeThreshold.Position   := defaultEdge;
      mainform.fpn.text                 := inttostr(defaultfpn);
      mainform.defects.text             := inttostr(defaultdefects);
      ResetAll(TRUE);
      OrigArray                           := PixelArray;
      DoResetAll                          := FALSE;
    end;

end.
```

Appendix B

AER Communication Source Code (Firmware)

```
/* ************************************************************************
 Program : ORASIS AER Readout Firmware (for PIC18F4620) in Microchip C18
  Module : main.c
    Date : See File Timestamp
  Author : Timothy G Constandinou
 Company : Imperial College London
 ************************************************************************ */

#include "p18f4620.h"   /* for TRISB and PORTB declarations */

#include <delays.h>
#include <portb.h>
#include <adc.h>
#include <usart.h>
#include <stdio.h>

#pragma config OSC=HS,BOREN=OFF,WDT=OFF,MCLRE=ON,LPT1OSC=OFF,PBADEN=OFF,
               LVP=OFF,DEBUG=OFF,XINST=OFF

#define max_events 750

#pragma udata big1
  char x[max_events+1];
#pragma udata big2
  char y[max_events+1];
#pragma udata big3
  unsigned int t[max_events+1];
#pragma udata

void initialise(void)
  {
    ClosePORTB();
    CloseADC();

    ADCON1 = 0b1111;
    TRISA = 0b11111111;
    TRISB = 0b01111111;
    TRISC = 0b10001110;
    TRISD = 0b00000000;
    TRISE = 0b00000000;
  }

void display_led(int led, int mode)
  {
    if (led==1) PORTCbits.RC4=mode;
    if (led==2) PORTCbits.RC5=mode;
  }

void display(int value) // To display a value (under 1023) on LED display
```

```
  {
    unsigned char value2=0;

    while (value>255) {value-=256; value2++;}
    PORTD=value;
    PORTE=4+value2;
  }

void orasis_reset(int mode)  // To assert a GLOBAL_RESET signal on ORASIS
  {
    PORTCbits.RC0=mode;      // Set reset
    PORTBbits.RB7=1;         // Set acknowledge high (active low)
  }

void orasis_flush_aer(void)  // Ensure AER registers are empty before starting
  {
    unsigned int n=0, m=0;

    display_led(1,0); display_led(2,0);

    while (n<50000)          // If no chip request for 50K ops then flushed
      {
        if(PORTBbits.RB6==0) // If chip request then acknowledges
          {
            PORTBbits.RB7=0; while(PORTBbits.RB6==0);
            PORTBbits.RB7=1; m++; n=0;
          }
        n++;
      }

    if (m>0)
      fprintf(_H_USART, "ORASIS_P2 AER Output: Flushed %d events...\n\r", m);
    display(m); Delay10KTCYx(500); // Displays number of events flushed
  }

void orasis_sample_aer(void) // Runs chip, samples AER and outputs to RS232.
  {
    unsigned int n, event=0, time=0;

    char *x_ptr = &x[0];
    char *y_ptr = &y[0];
    short long *t_ptr = &t[0];

    display_led(1,1); display_led(2,1);

    while(n<=max_events)t[n++]=0;
    while ((time<62500)&(event<max_events)) // Samples until buffer is full
      {                                     // or no activity for 500ms.
        if(PORTBbits.RB6==0) // If chip request then acknowledges
```

```
          {
            x[++event]=PORTA&0b00111111;
            y[event]=PORTB&0b00111111;
            t[event]=time;

            PORTBbits.RB7=0; while(PORTBbits.RB6==0); PORTBbits.RB7=1;
            time+=3;
          }
        time++;
      }

    display_led(1,0); display_led(2,0); display(event);

    if (event>0)
      {
        fprintf(_H_USART,
          "ORASIS_P2 AER Output: Streaming %d events...\n\r", event);
        for(n=1;n<=event;n++)
          fprintf(_H_USART, "Event %u at t=%u: %u,%u\n\n\r", n, t[n], x[n], y[n]);
        fprintf(_H_USART, "\n\r");
      }
    Delay10KTCYx(500);
  }

void main (void)
  {
    int value;
    unsigned char ctrl1,ctrl2,ctrl3;

    initialise(); orasis_reset(1);

    OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &
              USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH, 10);

    display_led(1,0); display_led(2,0);

    for (value=0;value<=1000;value++)
      {
        if (((value/16)%2)==0) {display_led(1,0);display_led(2,1);}
        if (((value/16)%2)==1) {display_led(1,1);display_led(2,0);}
        Delay10KTCYx(1); display(value);
      }

    orasis_flush_aer(); orasis_reset(0); orasis_sample_aer();
    orasis_reset(1);
    CloseUSART();
  }
```

Appendix C

System Simulation Schematics

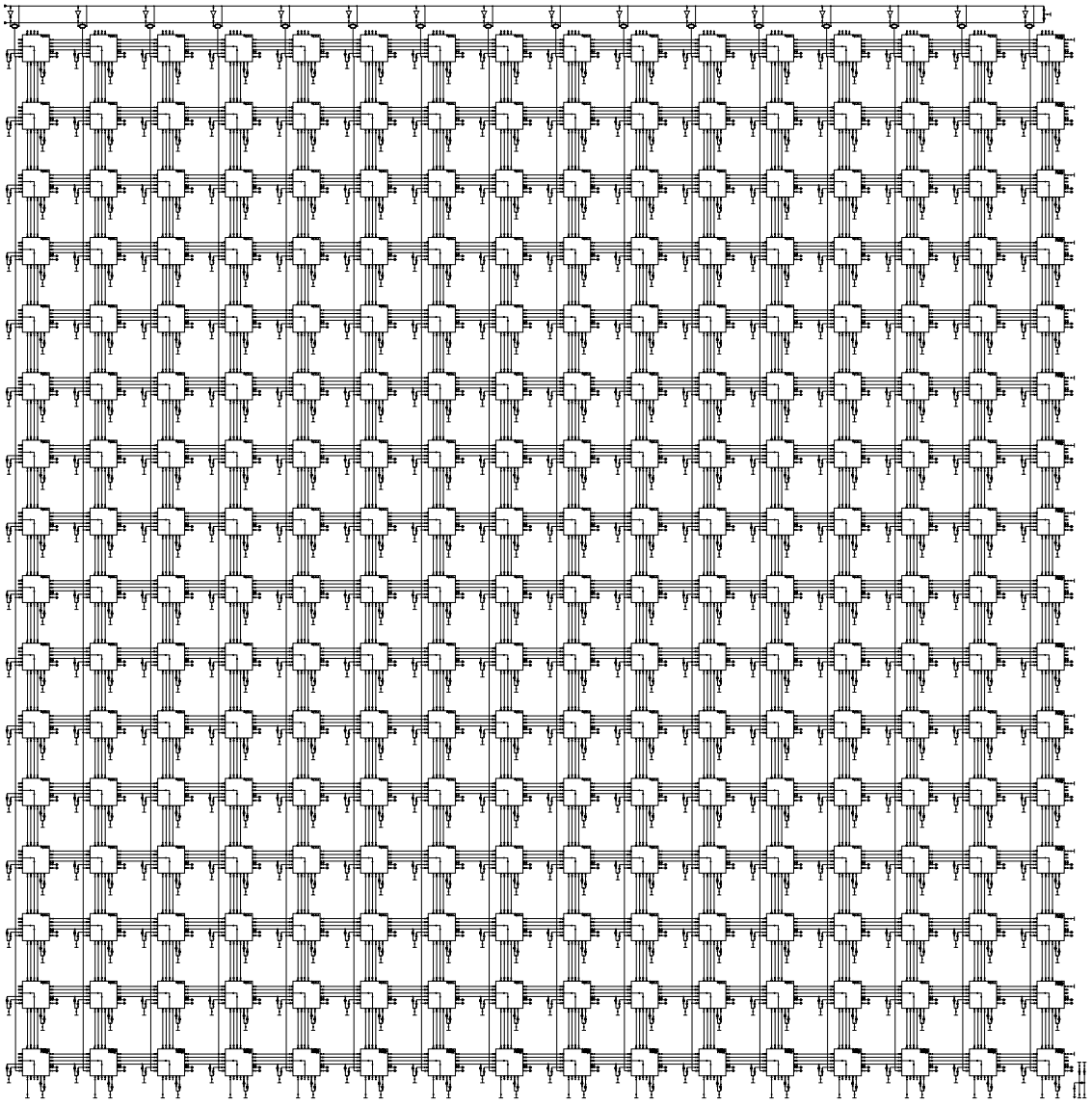


Figure C.1: Schematic diagram of the simulated 16×16 ASP array.

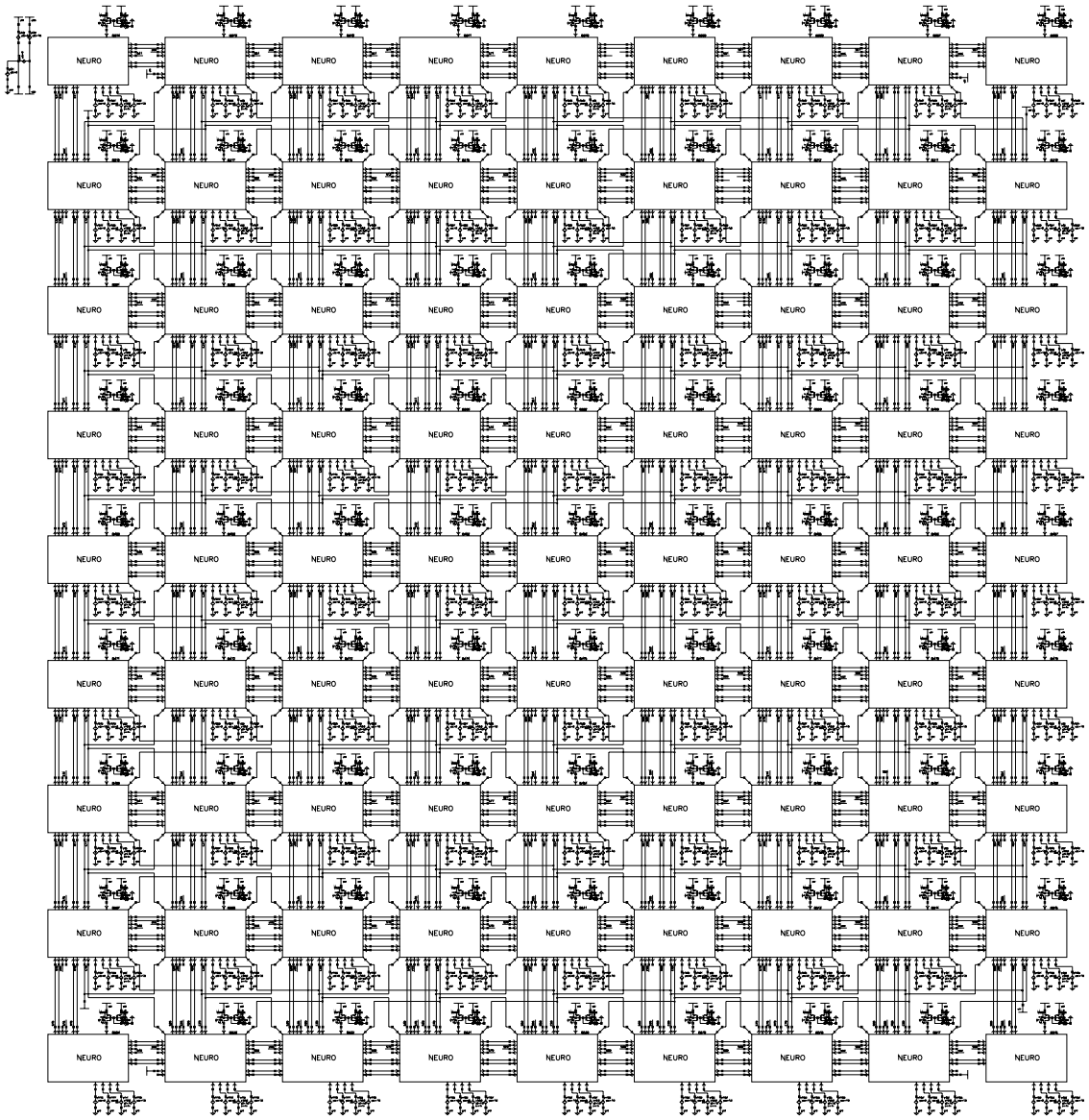


Figure C.2: Schematic diagram of the simulated 9×9 ABP array.

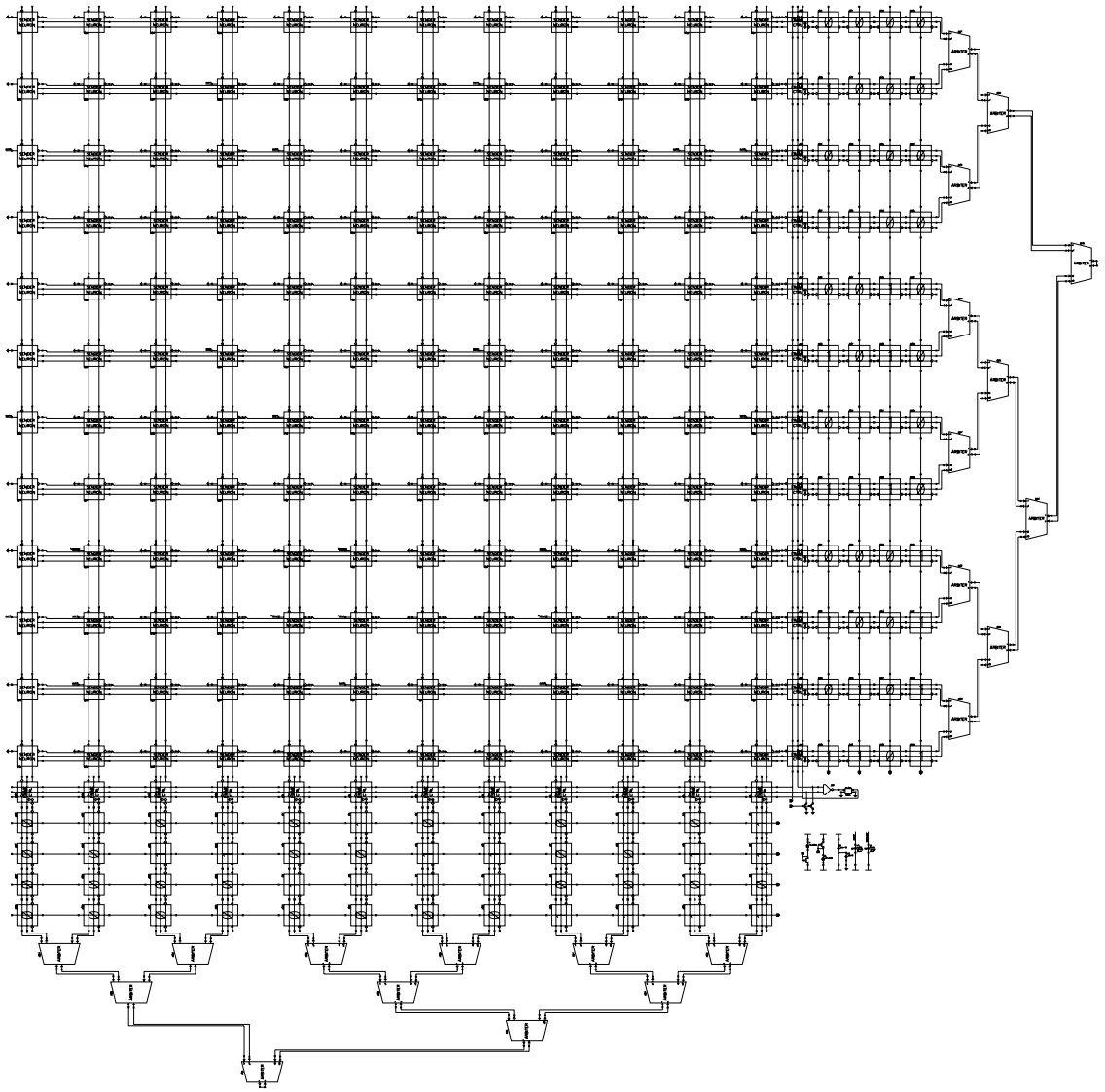


Figure C.3: Schematic diagram of the simulated 12×12 AER architecture.

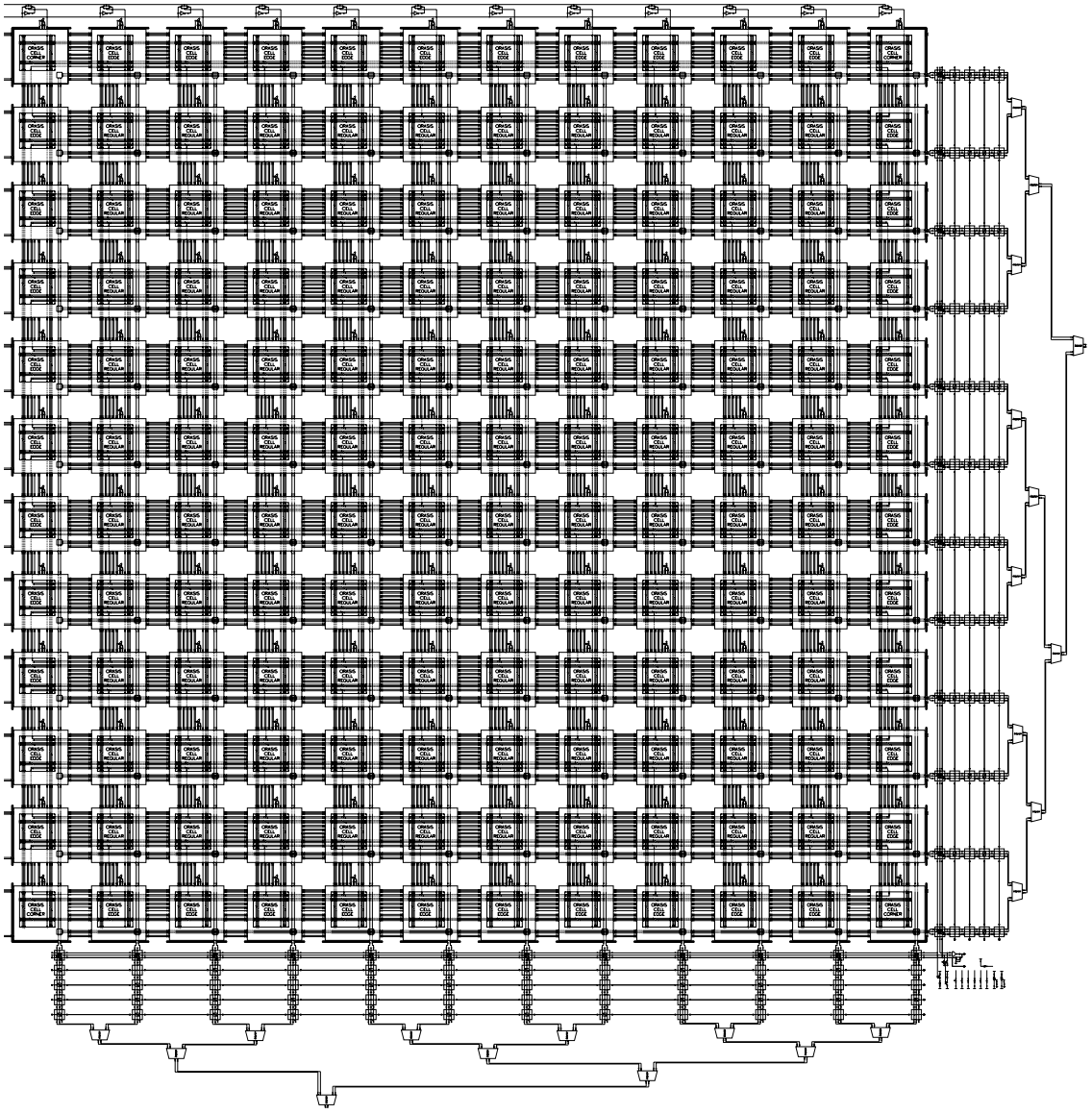


Figure C.4: Schematic diagram of the simulated 12×12 array.

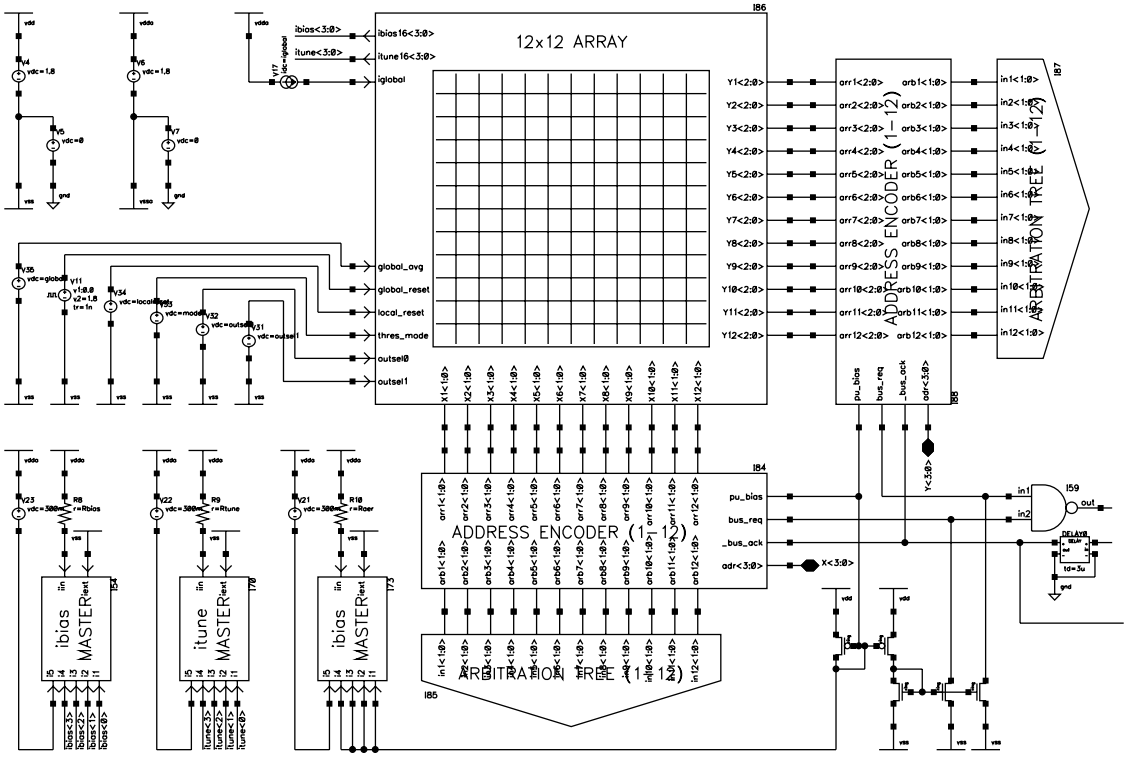


Figure C.5: Schematic diagram of the simulated system including a 12×12 distributed processing array.

Appendix D

Testboard Hardware

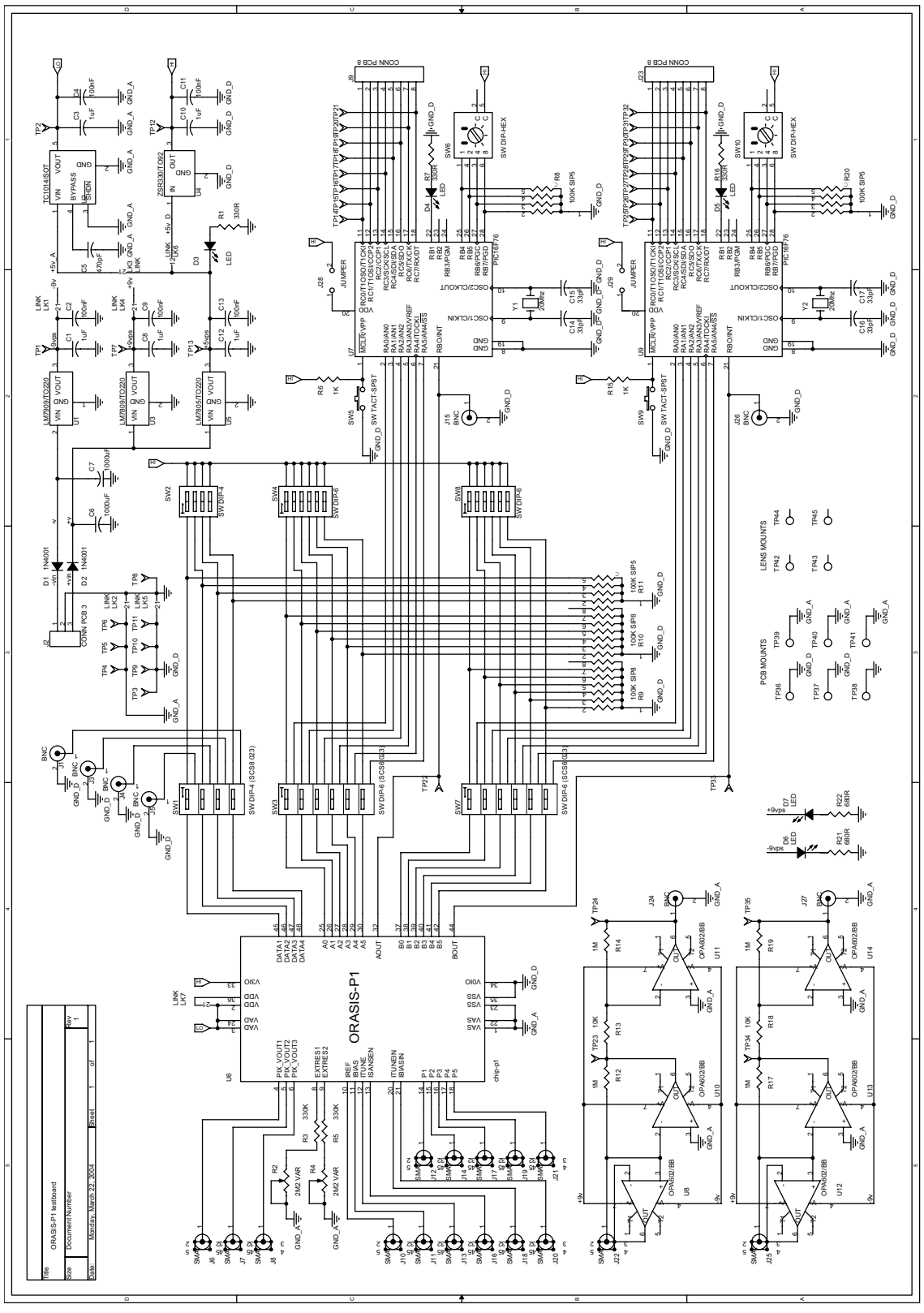


Figure D.1: Schematic diagram of the ORASIS-P1 platform for sub-circuit test and photodiode characterisation.

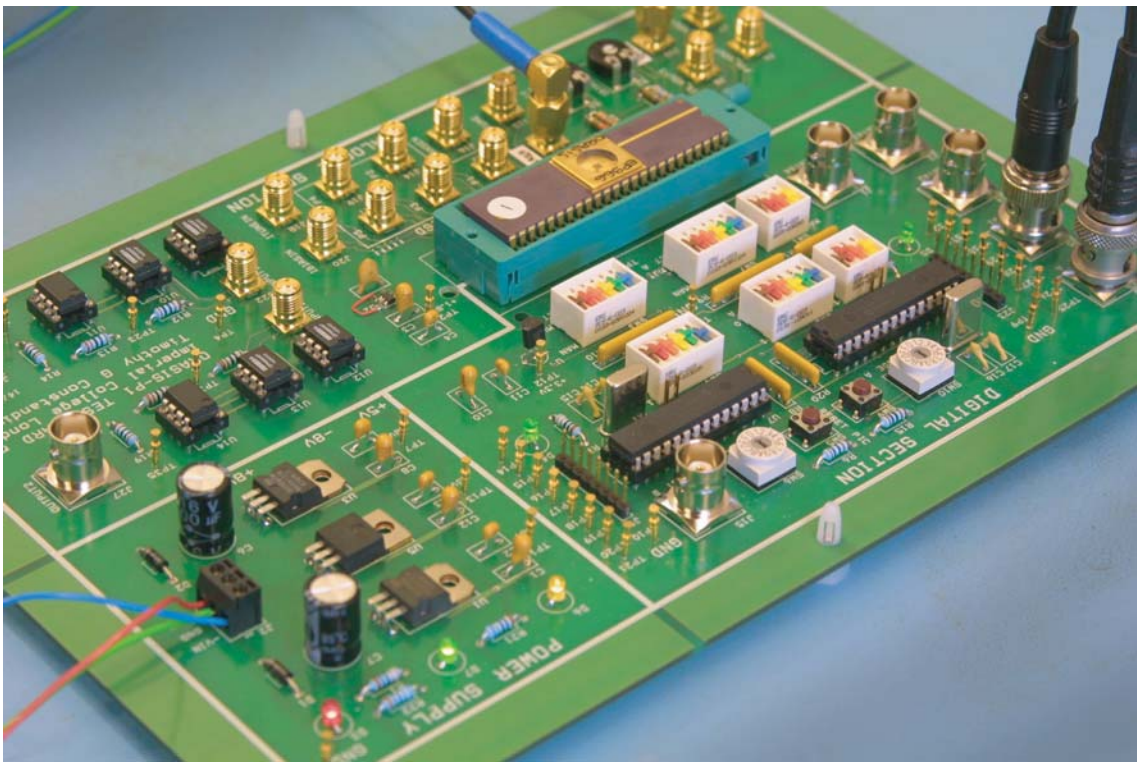


Figure D.2: Photograph of the ORASIS-P1 platform for sub-circuit test and photodiode characterisation.

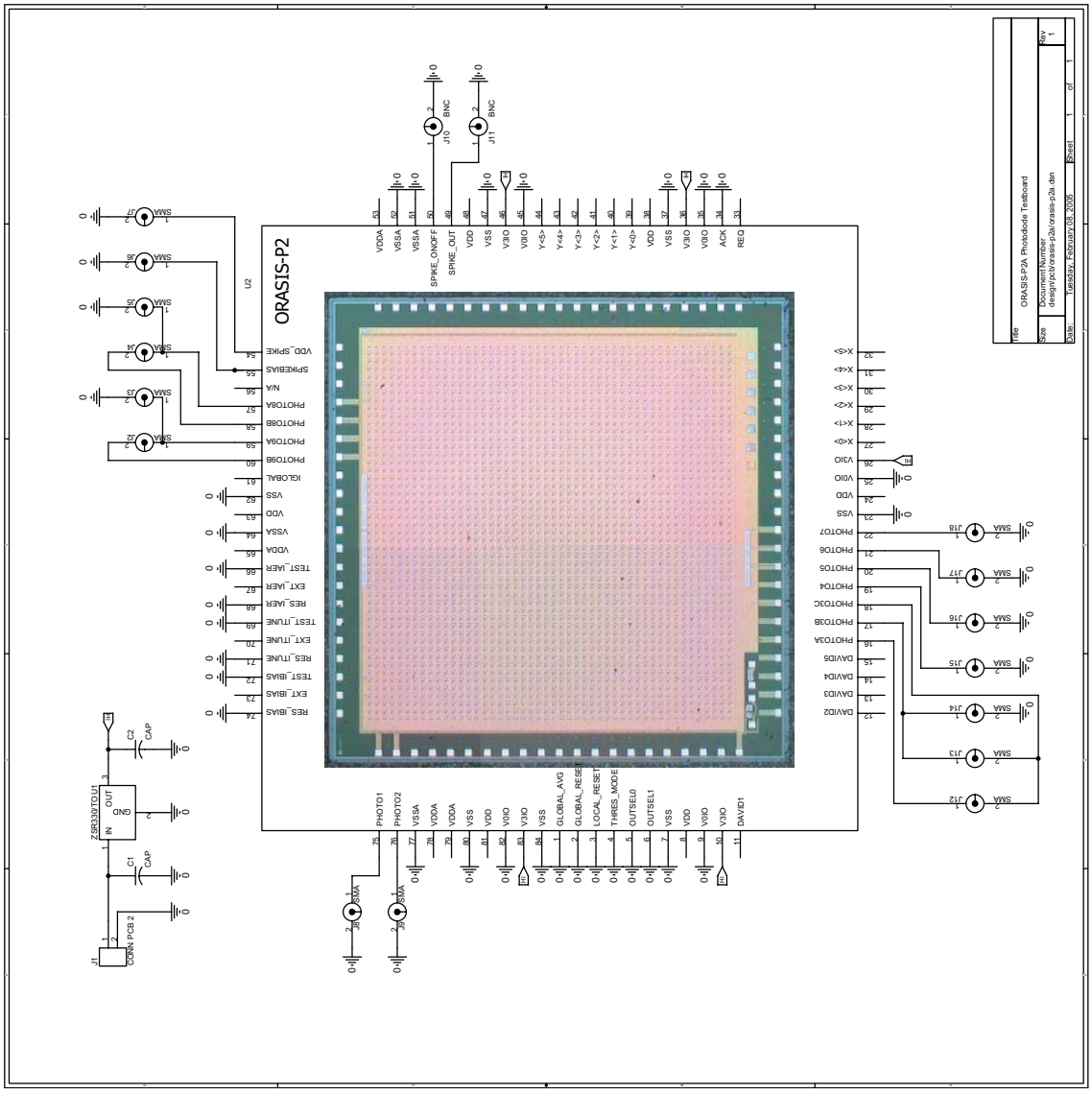


Figure D.3: Schematic diagram of the ORASIS-P2 platform for photodiode characterisation.

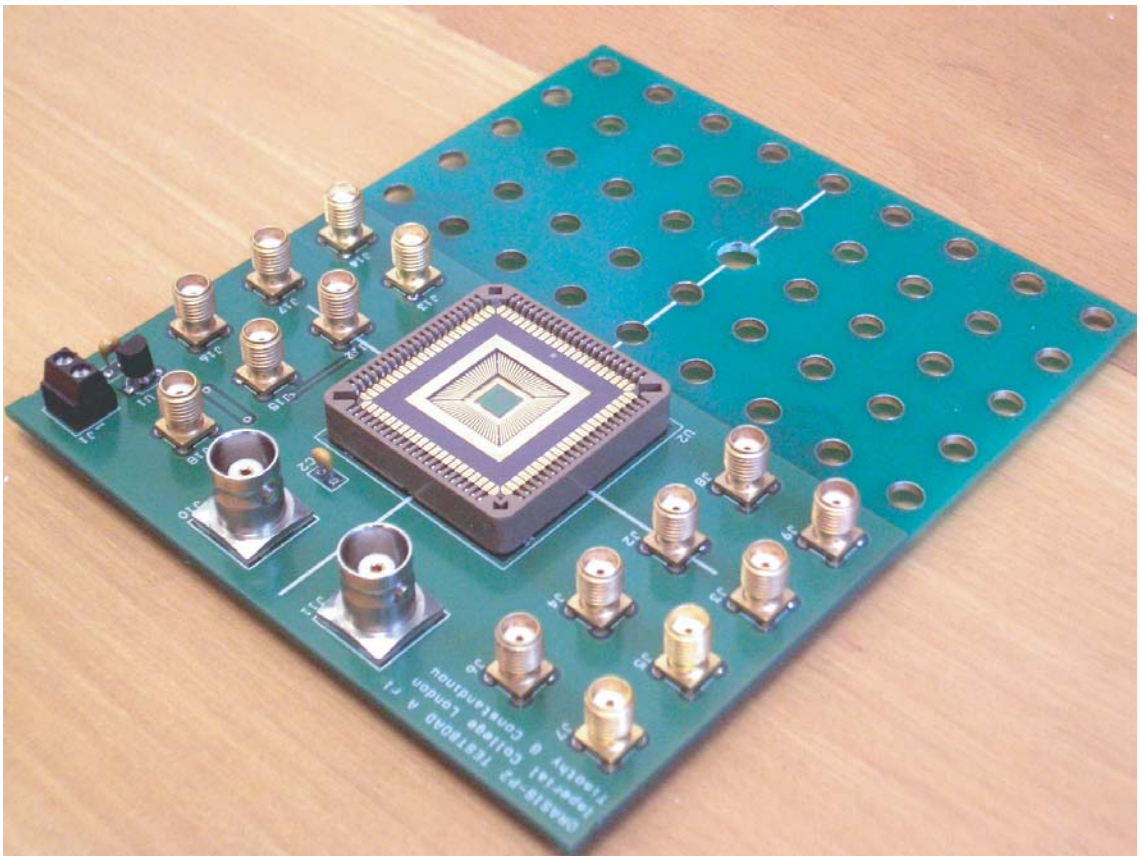


Figure D.4: Photograph of the ORASIS-P2 platform for photodiode characterisation.

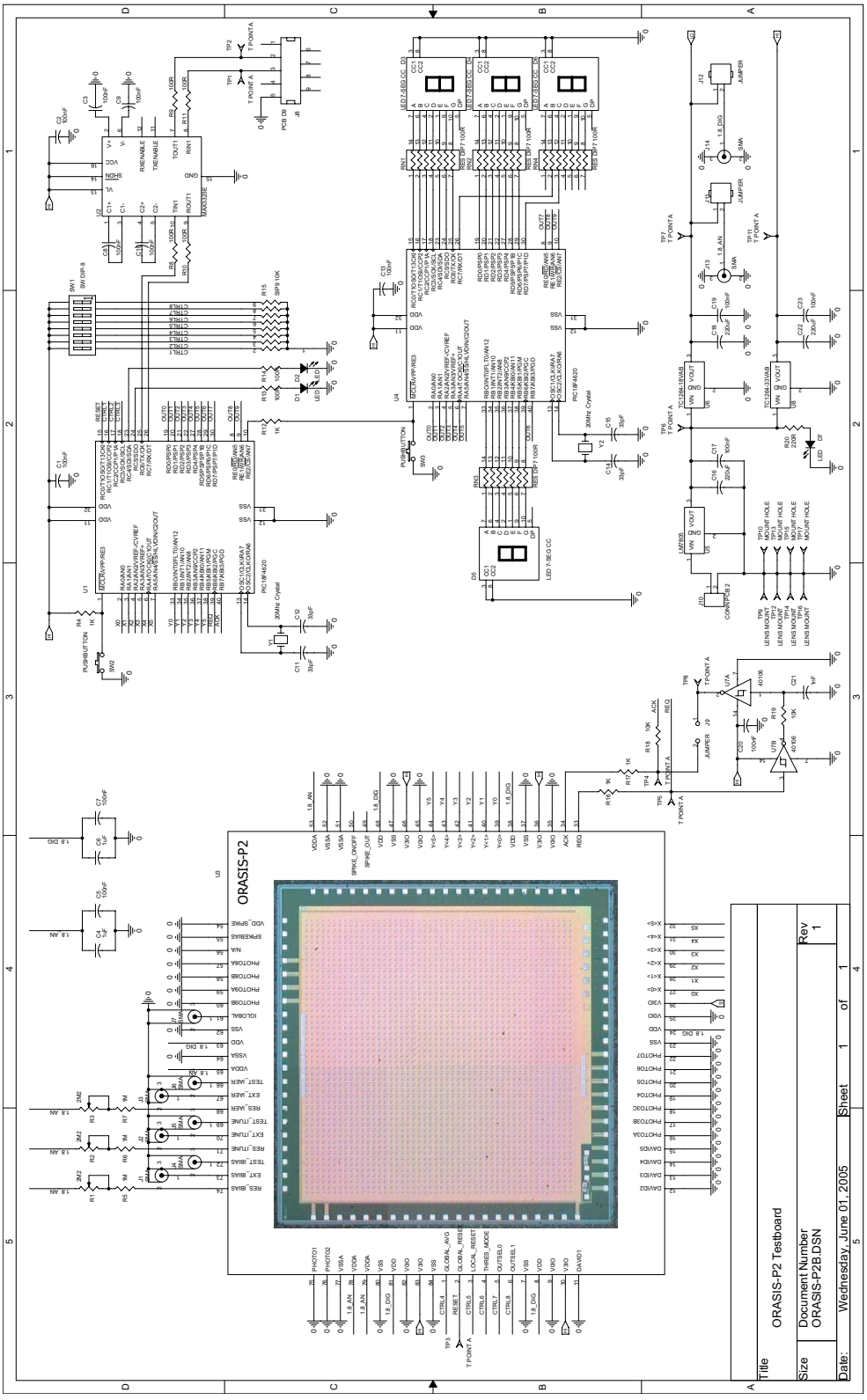


Figure D.5: Schematic diagram of the ORASIS-P2 platform for system test and validation.

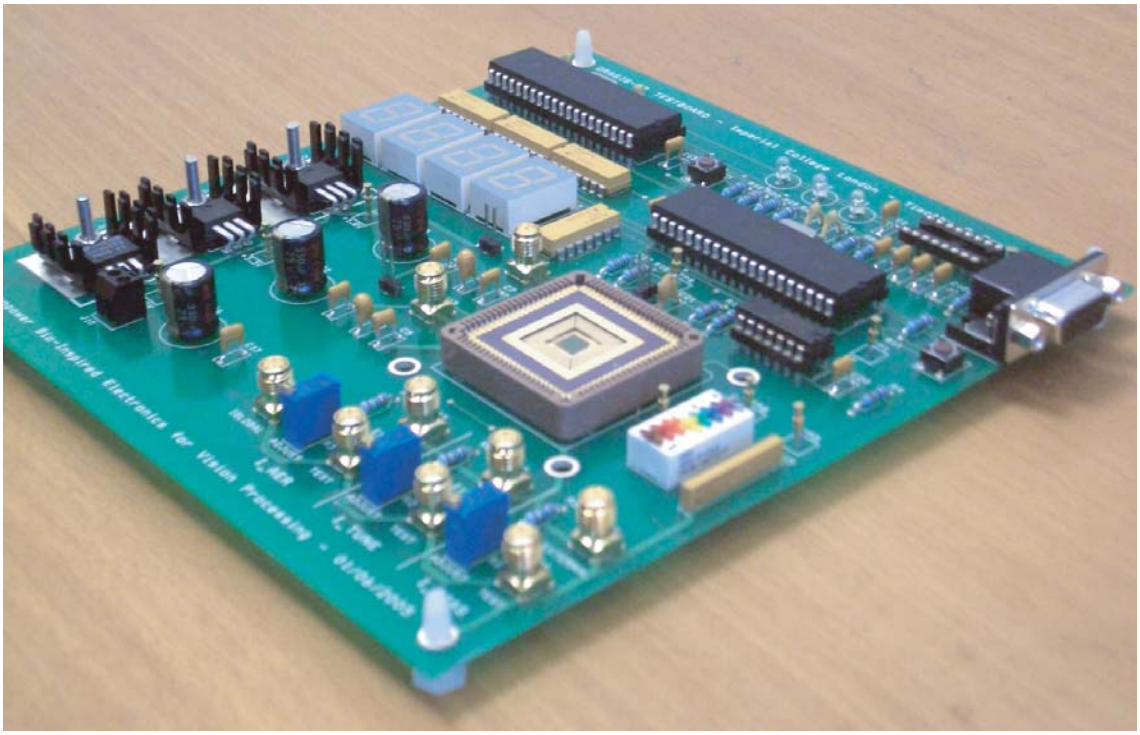


Figure D.6: Photograph of the ORASIS-P2 platform for system test and validation.

Appendix E

Optoelectronic Measurement Setup

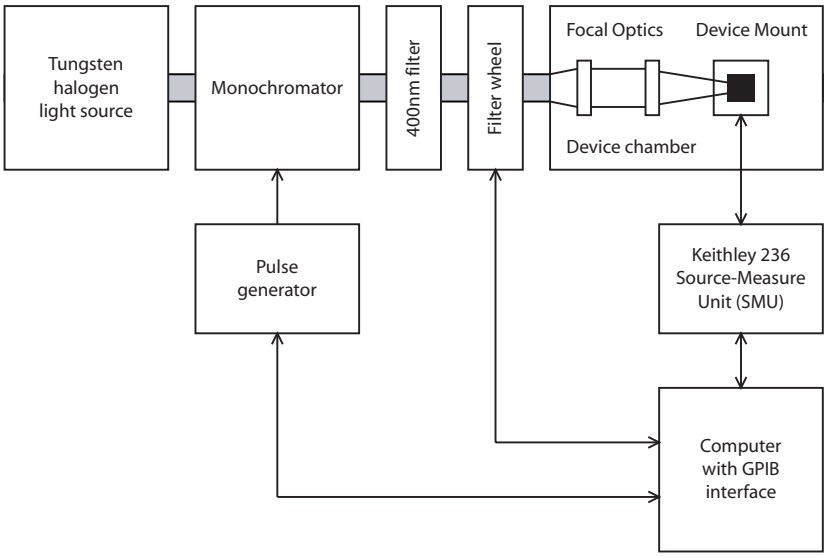


Figure E.1: Diagram of equipment setup for photodiode characterisation.

The equipment setup[1] for measurement of electrical (IV characteristics) and optical (spectral, responsivity, quantum efficiency, etc) is illustrated in Fig. E.1.

The tungsten halogen light source is UV filtered to remove sub-400nm wavelengths and avoid 2λ interference. This and the monochromator are characterised through the test optics with a calibrated photodiode (Newport 818-UV#3310). This provides the total incident light power for each monochromator (wavelength) setting at the device mount. Furthermore, a single small area photodiode is scanned across the incident light spot to determine its intensity profile. This characterisation is illustrated in Fig. E.2.

For each test device, the following measurements are taken:

- Electrical characterisation: At a set wavelength and calibrated light intensity the Source Measure Unit (SMU) sweeps device bias and measures corresponding photocurrent. Repeated for all Neutral Density (ND) settings (0, 0.15, 0.30, 0.41, 0.45, 0.77, 0.87, 1.10, 1.34, 1.47, 2.04, 5.06) and for under dark (no input) conditions.
- Spectral characterisation: At each wavelength the photoresponse is taken for various values of reverse bias. From this spectral photoresponse data, the responsivity and therefore external quantum efficiency can be determined.

[1]Optoelectronic measurements taken at facilities provided in The Blackett Laboratory, Department of Physics, Imperial College London.

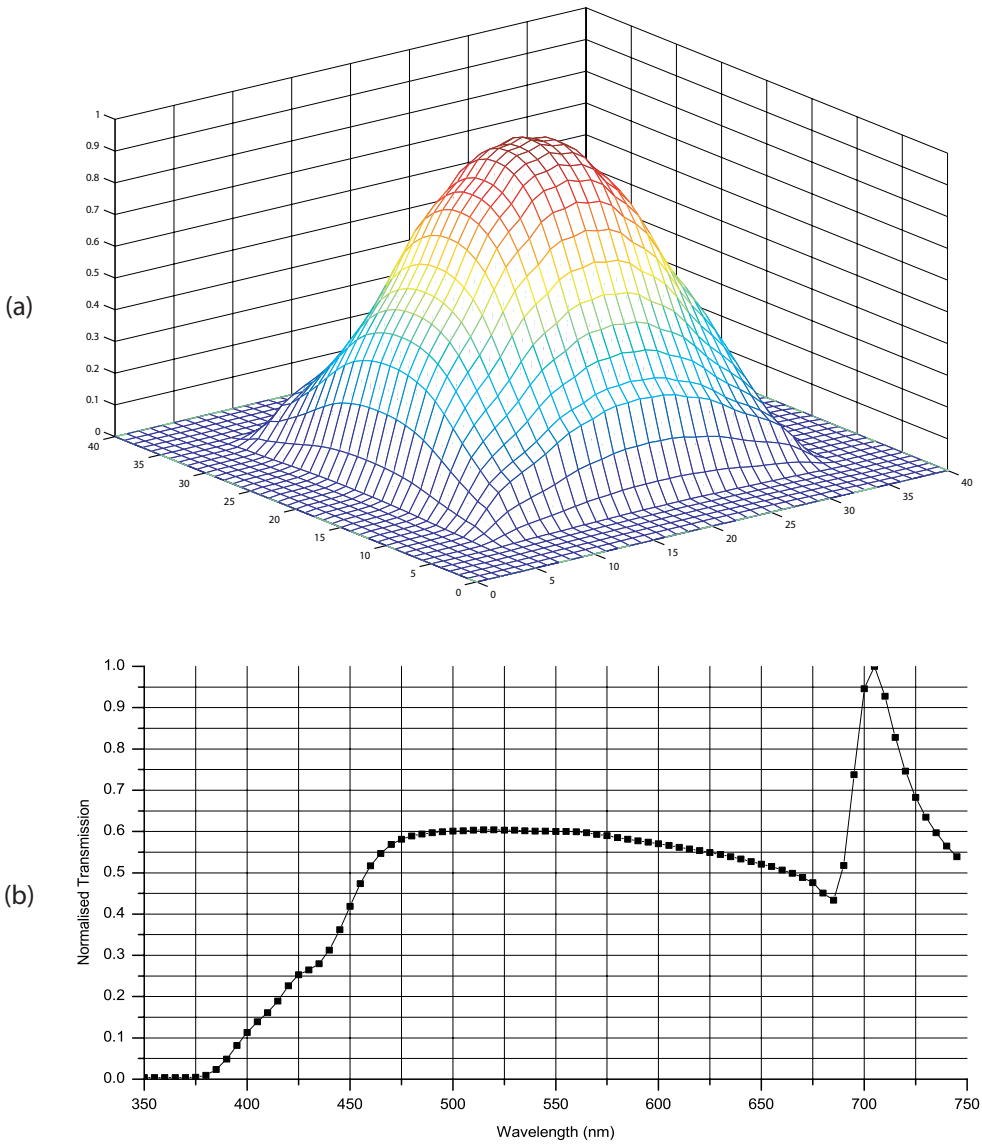


Figure E.2: Calibrated (measured) light source intensity characteristics. Illustrated are: (a) intensity profile and (b) spectral transmission (normalised to maximum value).

Appendix F

Publications

- T. G. Constandinou and C. Toumazou, "A Micropower Centroiding Vision Processor," *IEEE Journal of Solid State Circuits*, submitted, 2005.
- T. G. Constandinou, P. Degenaar and C. Toumazou, "An ON-OFF spiking photoreceptor for adaptive ultrafast/ultrawide dynamic range vision chips," *Proceedings of the IEEE Workshop on Biomedical Circuits and Systems*, S1/6-9, 2004.
- T. G. Constandinou, J. Georgiou and C. Toumazou, "A Nanopower Tuneable Edge Detection Circuit," *Proceedings of the 2004 IEEE Symposium on Circuits and Systems*, Vol. 1, pp. 449–452, 2004.
- T. G. Constandinou, J. Georgiou and C. Toumazou, "Towards a Bio-inspired Mixed-signal Retinal Processor," *Proceedings of the 2004 IEEE Symposium on Circuits and Systems*, Vol. 5, pp. 493–496, 2004.
- T. G. Constandinou and C. Toumazou, "Neuromorphic electronics for real-time biomedical image processing," *INE The Neuromorphic Engineer*, Vol. 1, No. 1, pp. 7, 2004.
- T. G. Constandinou, J. Georgiou and C. Toumazou, "Nano-power mixed-signal tunable edge-detection circuit for pixel-level processing in next generation vision systems," *IEE Electronics Letters*, Vol. 39, No. 25, pp. 1376-1377, 2003.
- T. G. Constandinou, T. S. Lande and C. Toumazou, "Bio-pulsating architecture for object-based processing in next generation vision systems," *IEE Electronics Letters*, Vol. 30, No. 16, pp. 1169-1170, 2003.
- T. G. Constandinou, J. Georgiou and C. Toumazou, "An Auto-input-offset Removing Floating Gate Pseudo-differential Transconductor," *Proceedings of the 2003 IEEE Symposium on Circuit and Systems*, Vol. 1, pp. 169-172, 2003.