A Compact System for Self-Motion Estimation

Thesis by

Ania Mitros

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2006
(Defended February 15, 2006)

Acknowledgements

This thesis would not be without Christof Koch, in whose lab I started my days at Caltech and its Computation and Neural Systems option. He continued to support me even when our primary research interests drifted apart and provided valuable editing of this manuscript. I am beholden to Chris Diorio who welcomed me into his lab at the University of Washington, provided me with resources, and shared some of his wisdom. The majority of the work presented herein was done in Chris's lab. I am lucky to have had two advisors who are not only intelligent, but who also deeply care about people and doing what is right.

I grew under the guidance of Oliver Landolt, an excellent analog circuit designer, from whom I learned much about engineering and project management. While we worked on the Vibrating Retina, he provided me with the most consistent mentoring of my PhD thesis.

My years at Caltech were more educational, fulfilling, and fun thanks to the other Forever-First-Years. We entered CNS together, took classes together, studied for candidacy together, and supported each other through some of the big bumps of graduate school. I hope the friendships I have out of those years last. Thanks to Reid Harrison, Theron Stanford, Alberto Pesavento, and Chuck Higgins for the engineering I learned in my early days at Caltech. Amish, I still think fondly of our days at the Aralia house.

Jeremy Holleman, Seth Bridges, Kambiz Rahimi, Jaideep Mavoori, and Miguel Figueroa were excellent technical resources and great folks with whom to chat and share a lab at the UW. Thanks to Ed Lazowska for his listening, understanding and advice. I am grateful to UW's CSE department for graciously hosting my first public art installation.

A PhD is a long road, and many people contributed to my happiness and balance during that time. My parents, Jozef and Katarzyna (or Tata and Mama), always encouraged my curiosity and did all they could to further my education. I could always rely on my brother, Piotr Mitros, and virtual sister, Jana Edelbrock, for advice and unconditional support. Thanks also to Piotr for chatting about circuits and building with discrete components. Thanks to Tata for answering my questions on processing, device physics, and floating gate transistors. I couldn't have kept my sanity without the love and wise advice of many good friends: Bjorn Christianson, Cameron Etezadi, Matt Knapp, Heidi Kneller, Seth LaForge, Ofer Mazor, Chris Simison, and others.

Abstract

Self-motion estimation is a vital problem for autonomous robots, frequently and appropriately addressed by vision algorithms. Most approaches involve repeating some local calculation over the entire imaging array, such as detection of locally salient features. A simple and local calculation can be efficiently implemented on the same chip as the photo-sensing array, thus parallelizing a huge computational task and vastly reducing the amount of data to transmit off chip. Mismatch between devices has previously been a stumbling block to producing truly useful arrays of local processing elements. Floating gate technology is used here as a compact means of programming away offsets in subcircuits to remedy this problem. A custom analog chip performs the above functions. For each pixel, the chip outputs sensed light intensity, the values of the vertical and horizontal intensity gradients, and a binary value indicating whether a feature is centered on that pixel. These values can be used as inputs to a motion estimation algorithm implemented on a standard computer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Self-motion estimation is a vital problem for autonomous robots. To move through an environment effectively and without harm to oneself, a robot must possess some means of estimating its motion relative to surrounding objects. The problem is constrained by the need for real-time operation, a compact solution that constitutes only a fraction of the robot's payload, and limited power availability. This thesis presents part of the solution, specifically circuits for image and feature detection, to be followed by digital processing for ego motion estimation. The system comprises a chip containing a photodetector array as well as local processing circuitry within the array. For each pixel, the chip outputs the detected light intensity, horizontal and vertical gradients at the current and previous time steps, and whether or not the pixel is centered on a salient feature. Off-chip digital postprocessing can be used to estimate the local 2-D motion at each salient pixel. The local motion estimate then can be combined to calculate the 3-D motion of the imager relative to its environment. The postprocessing could easily be integrated onto the photo-sensing chip.

1.1 Vision for Motion Estimation

A variety of sensory modalities can be used for motion estimation. The choice of one over another depends on the system and the environment in which it is to operate. For tracking absolute self motion, inertial sensors that directly measure linear or rotational acceleration may be the right answer. Inertial sensors may be implemented as MEMS (Micro Electro-Mechanical Systems), while necessary postprocessing can be embedded in either an analog or a digital circuit fabricated in CMOS (Complementary Metal-Oxide-Semiconductor). Because of the difficulties of integrating MEMS and CMOS on the same die, such integration can be solved by either a two-chip solution, or a more expensive bonding of the two dies into one package. Depending on the noise levels within the sensor and required precision of measurement, the sensor may need to be recalibrated to known landmarks by some secondary sensing scheme.

Many mobile robot applications also require knowledge of motion relative to the environment and motion of nearby objects. This is particularly true in the case of airborne or underwater robots, where motion relative to their medium may be different from the absolute motion measured by an inertial sensor. Also, inertial sensors cannot be extended to assess the approach of another object or vehicle. Medium- and long-distance sensors that are able to deal with such tasks include both vision sensors that detect light reflected by objects in the environment, and active sensors that emit and detect their own signal such as laser range finders. Of these, the preferred choice is dependent on the application. Benefits of active sensors include some independence from ambient lighting conditions. Active sensors are not appropriate for environments where stealth is important, since the signal emissions could be detected by other agents. Active sensors also require additional power to emit their signal. Power constraints thus limit the range of an active sensor since it becomes impractical to illuminate too distant an object.

Therefore, a number of applications are best served by passive vision sensors, for example, an autonomous mobile robot with limited power resources and the need to detect both ego-motion and motion or location of nearby objects. Although the system presented here does not calculate the trajectories of other moving objects, it provides a basis to do so in the future.

1.2 Custom Single-Chip Approach

Image processing systems, whether human, animal, or robotic, extract meaning from the world by building up an internal representation. Most such systems do this by hierarchically merging the information from individual photodetectors into simple features and processing those simple features to extract more abstract information. The simple features may be spatial, such as edges, salient local contrast patterns, or patches of color. They may be temporal, such as patches of pixels whose intensity changed significantly over some time interval. The type of simple features and the number of processing steps between these features and the desired abstract information will vary with the application and medium of implementation.

To determine the locations of all features in a given image, the same computation must be applied to each local image patch. This is a very computationally intensive processing step. In one implementation, for example, extracting the features required 20 times more computations than tracking the features as they moved from one image frame to the next, and 50 times more computations than estimating the camera motion [32]. Low-level computations require the most resources since they need to be repeated so many times. In some applications, the easiest ways to handle this computationally intensive problem are to use a fast modern CPU or to accept slower than real-time operation. However, for autonomous robotics applications, neither of these may be acceptable. Mobile robots are constrained by battery life and benefit from lower-power processors. They are constrained in the size of the payload that they may carry. Nonetheless, they need real-time control as they move through their surroundings. Likewise, the emerging field of ubiquitous computing desires small, cheap, low-power processors for embedding throughout an environment. In both these scenarios, embedding the most expensive computation in a specialized, lower-power processor can be a big win. Since the computation of features is a local computation, it lends itself well to a focal-plane implementation where the computational elements are located within the pixel array and process information in parallel.

1.3 Analog versus Digital Processing

The output of a pixel array requires substantial processing before a useful quantity like speed or direction of motion can be extracted. People argue whether analog or digital systems are better suited for information processing [68]. The underlying and more general question is *when* (or if ever) during signal processing should one switch from the analog to the digital domain. At a macroscopic scale, variables of interest are fundamentally continuous: volume or frequency of a sound, velocity, acceleration, light intensity[1] or wavelength, force, etc. Likewise, the sensors designed to measure these variables are analog. Proponents of digital systems cannot argue against the inherent existence of analog signals, but rather, for the conversion from analog to digital to occur as early as possible in the processing scheme.

The merits of analog and digital cause them each to be preferable in different situations. Since, typically, digital platforms can be reprogrammed more easily than custom analog chips, digital processing is a reasonable choice for parts of a system that may need to be modified, either as an algorithm is developed or refined or to accommodate a different set of requirements. On the other hand, analog processing is appropriate for initial amplification and filtering of a signal. It is well-suited to performing well-understood transformations known to be useful and unlikely to require modification. Analog implementation may result in a more compact and power efficient system, although the design process is more demanding. When applied to arrays of elements, due to connectivity limitations, analog processing tends to befit local computation and is less suitable for global analyses. The precision of the computation is limited by noise and distortion in analog systems, and by the number of bits chosen in a digital implementation. Increasing the number of bits requires more silicon area or processing time, and requires more power.

In general, then, local early processing that needs to be applied to every element of an array lends itself well to analog processing. If high precision is not required and the designer finds a means

[1]Technically, if we delve down to the level of quanta of light, we will be forced to admit the non-continuous, quantized existence of photons. However, in typical robotics applications, we collect so many photons per unit time that the quantity of interest is a continuous one, namely light intensity, rather than a discrete photon count. Even under dim lighting conditions when a single photon may be a significant fraction of the total count, it is more likely that we are interested in the average or expected value–a continuous value, possibly fractional when expressed as a photon count.

to embody the desired computation in a compact analog circuit, the resulting circuit may be faster (due to its parallelism), more compact, and less power hungry than an equivalent digital version. These traits describe exactly the vision application at hand. Detecting local features is a time-tested first step in many algorithms (e.g., [12, 29, 36, 43, 57, 70, 71, 72, 84]) so the loss of flexibility to easily reprogram that step of the computation is not likely to impact the final system.

1.4 Motion Estimation Algorithms

Several approaches to motion estimation exist in the literature. I will give a brief overview of some popular algorithms. The gradient model and energy models compute a value at each image location that combines information about how the image varies in space (the gradient) and time. The motion estimate is computed directly from the image. Token-based motion detectors select salient "tokens" or "features" and track these in time. The motion estimate is computed based on information about the location of these tokens, with the token detection as an intermediate step. This thesis uses a token-based algorithm.

Oftentimes, image detection is not continuous. Instead, the system acquires "frames" wherein a "frame" contains intensity information for one instant in time, and frames are typically separated by a constant time interval. Descriptions of the algorithms within this section use this notion of frames since, in practice, image processing algorithms are often implemented in this temporally discontinuous manner.

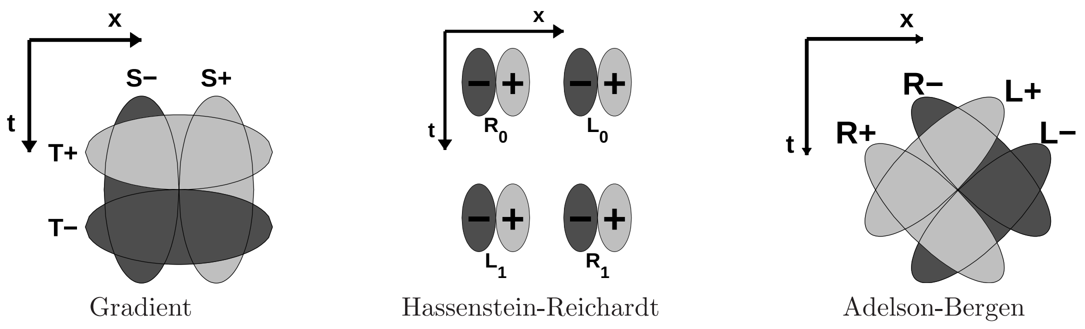
1.4.1 The Basic Gradient Model for Motion Estimation

Gradient-based motion methods are based on the approximation that local 2-D velocity can be calculated from the local gradient and from the change in local intensity over time [28, 41, 44, 45]. The basic gradient model computes the ratio of the temporal flicker and the spatial flicker, as illustrated in fig. 1.1. The temporal flicker is $\frac{\delta I}{\delta t}$, pictorially represented as $(T^+ - T^-)$. The spatial flicker is $\frac{\delta I}{\delta x}$, pictorially represented as $(S^+ - S^-)$.

We can derive the mathematical expressions for the model as follows. For a linear image, denote the image intensity at location x and time t by $I(x,t)$. Assuming that the brightness of image points does not change during the motion (i.e. $\delta I/\delta t = 0$), the image subjected to a translation at velocity v for a time interval τ is denoted by $I(x - v\tau, t + \tau)$. By the chain rule of basic calculus we then can write the 1-D velocity as:

$$\frac{\delta x}{\delta t} = -\frac{\delta I}{\delta t} \cdot \frac{\delta x}{\delta I}$$

To extend this to two dimensions, represent image intensity at location (x, y) and time t by $I(x, y, t)$.



$\frac{T^+ - T^-}{S^+ - S^-}$ $(R_0 \times R_1) - (L_0 \times L_1)$ $(R^+ - R^-) - (L^+ - L^-)$

Figure 1.1: Three motion models shown on a space-time plane, where t denotes time and x denotes space.

The same assumption of light constancy leads to the following for motion in two dimensions:

$$\begin{aligned}\frac{dI}{dt} &= \frac{\delta I}{\delta x} \cdot \frac{dx}{dt} + \frac{\delta I}{\delta y} \cdot \frac{dy}{dt} + \frac{\delta I}{\delta t} = 0 \\ &= \frac{\delta I}{\delta x} \cdot v_x + \frac{\delta I}{\delta y} \cdot v_y + \frac{\delta I}{\delta t}\end{aligned}$$

where v_x and v_y are the components of the velocity along the x-axis and y-axis, respectively. The set of velocities that satisfy this equation then lie on the line defined by (v_x, v_y) in velocity space. This formulation is somewhat problematic, however, in that a solution for velocity will necessarily involve the spatial gradient ($\delta I/\delta x$ or $\delta I/\delta y$) in the denominator. This is an ill-conditioned equation since the denominator can go to zero. More sophisticated implementations of the gradient model extend the equations, for example incorporating higher derivatives into the denominator. Taking multiple measurements, each being a different order derivative of the intensity, leads to the following set of linear equations:

$$\frac{\delta^2 I}{\delta t \delta x} = v_x \frac{\delta^2 I}{\delta x^2} \quad , \quad \frac{\delta^3 I}{\delta t \delta x^2} = v_x \frac{\delta^3 I}{\delta x^3} \quad , \quad \ldots \quad , \quad \frac{\delta^n I}{\delta t \delta x^{n-1}} = v_x \frac{\delta^n I}{\delta x^n}$$

A least-squares approach to approximate v_x produces [45]:

$$v_x = \frac{\sum_{k=2}^{n} \frac{\delta^k}{\delta x^k} I(x,t) \cdot \frac{\delta^k}{\delta x^{k-1}} \cdot \frac{\delta}{\delta t} I(x,t)}{\sum_{k=2}^{n} \frac{\delta^k}{\delta x^k} I(x,t) \cdot \frac{\delta^k}{\delta x^k} I(x,t)}$$

The terms in the denominator are various order derivatives of the intensity at a single point in the image. Since the derivatives are unlikely to *all* have values near zero, the equation should be well-conditioned.

1.4.2 Energy Models for Motion Estimation

Energy-based motion detectors (e.g. [3, 14, 38, 39, 78]) merge spatial and temporal information to compute the motion energy, as shown in fig 1.1. These models look for patterns in x-y-t space; a velocity corresponds to orientation in this space. In contrast to the basic gradient model which calculates speed directly (e.g., in pixels per frame), the output of these detectors indicates how closely the stimulus matches the tuned detector. A nice review of the basics of spatio-temporal motion detection can be found in [45].

The inspiration behind some of these models has been biologically driven. The Adelson-Bergen model [3] and that of Watson and Ahumada [78] are consistent with the electrophysiology of motion perception in primate cortical complex cells. The Hassenstein-Reichardt [38] correlation motion detector accurately models the optomotor response in flies and was later extended by van Santen and Sperling [76, 77] to human motion perception. The Barlow-Levick [8] mechanism describes direction selectivity in the rabbit retina.

Fig. 1.1 illustrates two of these models, the Hassenstein-Reichardt [38] and the Adelson-Bergen [3]. The Reichardt correlation motion model computes a correlation between two patches in time-space ($L_0 \times L_1$) and subtracts the result from the correlation between two other patches ($R_0 \times R_1$). For example, consider a high-contrast feature that stimulates R_0. If it moves in space and time to stimulate R_1, the correlation ($R_0 \times R_1$) will be high, and the overall response will be large. The pair of detectors L_0 and L_1 will respond to motion in the opposite direction from R_0 and R_1. Thus, the sign of the difference of the final output $(R_0 \times R_1) - (L_0 \times L_1)$ will indicate direction. The detector is velocity-tuned. The Adelson-Bergen model uses filter kernels oriented in space-time detect the presence of motion in a particular direction [78]. A black and white bar moving rightward will cause a large response in $R = R^+ - R^-$. Conversely, the same bar moving leftward will activate $L = L^+ - L^-$. Like the Reichardt model, the final output of the Adelson-Bergen motion detector is a comparison of motion in the rightward and leftward directions. In the Adelson-Bergen model, pairs of filters in quadrature phase are applied to images and the outputs are squared and added. The use of filters in quadrature phase ensures a response to both edge and bar stimuli. The squaring operation results in indifference to the sign of the changes.

1.4.3 The Kanade Motion Detector

Tomasi and Kanade [75] proposed a motion detector based on the assumption that local patches of an image are not distorted substantially between frames. This is true if the illumination is constant and if motion can be fully described by a translation. This notion is mathematically summarized as:

$$L(\vec{x}, t) \quad = \quad L(\vec{x} + \vec{d}, t + \tau) \tag{1.1}$$

where $L(\vec{x}, t)$ is an image patch at location $\vec{x}$ and at time t, $\vec{d}$ is the displacement (motion) between frames, and τ is the time between frames. The full formulation provides an algorithm for detecting salient features that are easily trackable, where trackability is rigorously defined by minimizing the error between the constraint in eqn. 1.1 and the actual change in image patches. I describe the mathematics in more detail in sec. 2.2. The Kanade algorithm provides a means of estimating the displacement $\vec{d}$ by inverting a matrix and multiplying it by a vector (see eqns. 2.3 and 2.4).

The Kanade detector and variations upon it are popular in vision literature. However, the computations are difficult to implement directly with analog circuits and computationally expensive in digital post-processing. Sec. 2.3.2 describes the difficulties encountered by Pesavento [61] in building analog circuits that were ultimately crippled by mismatch [62], while in sec. 2.3.5 I explain why the very non-linear nature of Pesavento's circuits would make it very difficult to fix his mismatch problems. Digital implementations require a sequence of computationally expensive operations. The analog image data first needs to be digitized by an A/D (Analog to Digital converter). The Kanade feature computation itself requires 3 multiplications to calculate the characteristic matrix, plus another two multiplications and one subtraction to calculate the featureness value, plus a thresholding operation (see sec. 2.2 and 2.4.3). Floating-point multiplications are especially expensive and in some computer architectures may require multiple clock cycles to compute. Even an integer multiplier requires approximately 40 times more layout area than a comparable analog multiplier with floating gates for mismatch compensation (see sec. B).

1.4.4 Token-Based Motion Estimation

Token-based (or feature-based) motion detectors [12, 33, 57, 65, 82, 85] compute local motion by selecting salient "tokens" or "features" to track between frames. They compare a target image patch in one frame to potential match candidates in another frame and select the closest match. This match provides an estimate of where the image patch moved between the frames. Attempts are made to perform correlations more selectively since performing a correlation between every pixel's neighborhood and every other pixel's neighborhood in another frame would be unmanageably expensive.

Consider an image of N pixels within which we would like to make comparisons between neighborhoods of K pixels. To compare the neighborhood of every pixel in one frame to the neighborhood of every other pixel in a successive frame, N^2 neighborhood comparisons are necessary. Assuming no limitations on memory size nor memory speed, this calls for N^2 multiplications, $N \cdot\ K \cdot (K-1)$

additions, and $N \cdot (K-1)$ comparisons. We also need to store in memory the N^2 floating-point results from each multiplication. A modern 2 GFLOPS (billion floating-point operations per second) processor could process images of up to 38×39 pixels at a frame rate of 30Hz. Token-based methods improve throughput and thus allow processing larger images.

Token-based correlation methods restrict correlations to patches centered on tokens (aka features), where "tokens" are defined [75] as image patches that are easy to recognize from frame to frame at multiple points in time. Computational resources thus are not wasted on image patches that are unlikely to be correctly and uniquely tracked, such as areas of low contrast.

1.4.5 Our Implementation

In all these motion estimation methods, some local calculation is repeated over the entire imaging array. If a multi-purpose digital computer is used, it implements these massively parallel operations serially. These low-level repeated calculations are thus a major hindrance in the implementation of real-time solutions [11, 32, 48].

The system presented herein implements a token-based correlation algorithm. I considered using Kanade features, extending the work of Pesavento [61], but correcting for mismatch in a circuit as complex as that necessary to calculate Kanade features is prohibitively expensive. Instead, the hardware embodies a simpler feature detector introduced in sec. 2.4. Each pixel measures the local light intensity. Based on its neighbors' outputs, it also calculates the local horizontal and vertical gradients. Each pixel is deemed to be centered on a salient feature if both the x-gradient and y-gradient are of sufficiently large magnitude. For those pixels deemed salient, a local search compares each given pixel to its neighbors to determine which neighbor most likely viewed that same feature at a previous time step. The gradient calculation is performed by an analog circuit embedded within every pixel. When a pixel is selected to be read out, its outputs are also directed to a saliency circuit (unique on the chip). The local search for a match in a previous frame is controlled by digital processing external to the chip. Thus, a compromise is struck to take advantage of the merits of both the digital and analog domains. The simple and local calculation is efficiently implemented on the same chip as the photo-sensing array, parallelizing a huge computational task and vastly reducing the amount of data to transmit off chip. Calculations requiring more complex addressing, namely the search for a most-similar neighbor, are performed by digital circuitry.

1.5 Motion Estimation in Hardware

This work is not the first to attempt to implement motion estimation in custom hardware. Multi-chip designs rely on a CCD camera as the front-end. The raw image data is serially read out to a multi-purpose digital postprocessor. Monolithic implementations integrate photodetection and information processing (often in the analog domain) on the same custom chip.

Multi-chip designs benefit from flexibility to reprogram the system. However, the serial nature of CCD readout and typical digital processor operation is poorly matched to the dramatically parallel nature of the image processing problem. Real time implementations thus rely on clever optimizations and modern hardware with fast clock rates. For a demonstration or application where size and power are not of concern, this is very appropriate. The target application of this thesis, namely mobile robotics, has more stringent requirements.

Monolithic implementations often parallelize the low-level image processing by including some of it within each pixel. Alternately, the processing is done by circuitry at the edges of the photodetector array but still on the same chip. If done well, focal-plane processing can reduce the amount of repetitive local computation required off-chip, and can also reduce the amount of data that must be transferred from one chip to another. Unfortunately, many previous single-chip designs emphasized biological inspiration rather than motivating functional reasons for design decisions, showed test data only for synthetic bar stimuli, failed to discuss the effects of using natural stimuli, and/or used a communication scheme (Address Event Representation, aka. “AER”) that has limited application to larger arrays than those built on the demo chips.

However, it is more edifying to focus on previous work that has chosen similar design priorities as this thesis than to rebuff those whose authors apparently had different priorities. The following pieces of work are most closely related to the work of this thesis. Fiore et al. [32] and Benedetti and Perona [11] both implemented in hardware motion algorithms based on the Tomasi-Kanade feature tracking algorithm [75]. The central algorithm of this thesis derives from the same source as theirs. Pesavento [61] implemented the Tomasi-Kanade detection algorithm on a single chip, but did not attempt to extend the system to complete a higher level task. Díaz et al. [23] implemented a gradient optical flow algorithm (in contrast to feature tracking), also on an FPGA.

Benedetti and Perona [11] developed some simplifications to reduce the computational complexity and facilitate real-time implementation. Benedetti’s Field Programmable Gate Array (FPGA) implementation did not perform feature tracking nor motion estimation; it only detected Tomasi-Kanade features. The interesting aspect of Benedetti’s work was that it showcased his system of multiple FPGAs. This is a worthwhile technique for solving problems too complex for a single FPGA. What is more relevant to this thesis is that the choice of feature detection as a representative task underscores the high demands of early vision processing and exemplifies the difficulty of accommodating an inherently parallel computation by an architecture that is itself not equally parallel.

Fiore’s implementation [32] consisted of an input video stream fed to an FPGA which performed all the necessary calculations. The FPGA performed feature extraction, feature tracking, and motion estimation. Fiore’s system achieved a throughput of 15 frames/sec for 256×256 8-bit images. The feature extraction accounted for 93.5% of the computational cycles. Parallelization of the feature extraction, as recommended in this thesis, is a worthwhile improvement on the design [31] because it decreases the computational demands placed on the digital postprocessor. This frees the multi-purpose digital hardware for use in other tasks, allows the use of a smaller cheaper digital processor, and makes possible the integration of digital postprocessing as an Application-Specific Integrated Circuit (ASIC) on the same die as the image sensing and feature extraction. It also permits motion estimation from a larger imaging array.

Pesavento's chip [61] successfully implemented the Tomasi-Kanade feature detector in local analog circuits embedded within every pixel. While the individual feature detectors worked well, it was impossible to find bias settings where an acceptably large percentage of the feature detectors would all function. Mismatch between elements rendered the array as a whole unusable [62] because the array elements had such dramatically different operating characteristics at any single bias setting.

Díaz et al. [23] implement the Lucas and Kanade [53] gradient optical flow algorithm in an FPGA. Díaz provides tables that allow an estimate of the number of gates used and thus the silicon area of the FPGA. If that same area of silicon were used for an analog pixel array, an imager of similar resolution performing the same calculation could be built but without the frame rate limitation of 30Hz. That is, the all-analog version using the same silicon area could run much faster.

This thesis presents a chip that is similar at a functional level to that of Pesavento, in that it implements feature detection within every pixel. However, mismatch is compensated by integrated programmable floating gates. The circuits have been completely redesigned due to this constraint. The chip is then combined with a digital postprocessor to extend the system to one akin to that of Fiore. The system presented here is more appropriate for mobile robotics applications than that of Fiore because it is, in principle, more compact and less power hungry. It is also easily scalable to larger array sizes simply by making the custom pixel array bigger, with minimal additional demands on the digital processor.

1.6 Mismatch Between Array Elements

Mismatch between devices has been a critical barrier to producing truly useful arrays of local processing elements. Imperfections in chip manufacturing result in circuit elements which are not quite identical. These disparities can render arrays of circuits unusable [62] when the array elements operate too differently. Standard approaches to mismatch reduction, such as using larger devices, surrounding critical devices with non-functional dummies, or mismatch invariant layout (sec. 3.1.2), require too much silicon area to be practical for embedding in every single element of an array.

Subthreshold design has been disproportionately plagued by mismatch. This is partly due to its widespread application in arrays of neuromorphic elements, where mismatch is difficult to counter as for any array. Additionally, the proportional change in current due to a given threshold offset may be greater for a transistor in subthreshold than for the same device above threshold, since subthreshold transistors display an exponential relationship between current and voltage while above-threshold transistors exhibit a square relationship. Mismatch has limited application of designs using subthreshold transistors, possibly more than any other factor [20].

The problem of mismatch is pronounced in analog focal plane processing arrays due to their use of logarithmic photodetectors. Integrating photoreceptors, which reset a photodiode and sample it some time later, are affected primarily by mismatch between the photodiodes. In modern high-quality processes, this mismatch is low enough to be acceptable for many applications. Measurements in 1.2μm and 2.0μm processes [46] indicate less than 2% mismatch between photodiodes. Unfortunately, because they are sampled, integrating photoreceptors are not well-suited to continuous-time analog computation. In contrast, logarithmic photoreceptors output a continuous-time voltage in response to illumination. Logarithmic photoreceptors send the photocurrent generated by the photodiode through a transistor. The transistor is operated in subthreshold and thus performs a logarithmic compression on the signal. Logarithmic compression is analogous to human vision and extends the dynamic range of the photoreceptor far beyond that of a CCD camera [51] or a CMOS integrating photoreceptor. Mismatch in logarithmic photoreceptors results primarily from voltage threshold mismatch in the transistor that performs the logarithmic transformation, and the mismatch between transistors is larger than that between photodiodes. Fig. 3.11 illustrates the effects of mismatch on a small array of logarithmic photoreceptors. The shift in output voltage due to mismatch is of a magnitude comparable to that resulting from typical signals. Some sort of mismatch correction, either within the photoreceptor array or in later processing, is critical to fabrication of functional logarithmic imagers [51, 61]. I present a modified photoreceptor with floating-gate mismatch compensation and compare it to a non-programmable photoreceptor in fig. 3.12. Sec. 3.3.5 quantifies the obvious improvement.

Other subcircuits within the pixel are also impacted by mismatch. Floating gate technology (described further in sec. 3.1.4) is used here as a compact means of reducing mismatch in both the photoreceptors and other thoughtfully-chosen subcircuits where mismatch could cripple the desired computation. Floating gates are a form of non-volatile memory, which means they are able to store their values for extended times (on the order of years) as well as when power to the circuit is turned off. Floating gate transistors are widely used in digital memory storage, namely in EPROM and EEPROM, whose applications include CompactFlash cards and the BIOS (Basic Input/Output System) of a computer. However, floating gate use in analog circuits remains within the toolbox of a much smaller group of designers. This thesis describes circuits which are an interesting addition to that toolbox. Floating gates are embedded in the logarithmic photoreceptor, in a novel difference circuit, and in the circuit which determines saliency of image features. The floating gates are programmed once to permanently remove mismatch between the subcircuits into which they are embedded.

1.7 Beyond Preprocessing

This thesis presents a focal plane aVLSI (analog Very Large Scale Integration) chip specifically intended for integration into a system which accomplishes a useful task: ego motion estimation. Focal plane processing has existed for years. Many interesting vision preprocessors have risen out of this field [7, 15, 17, 27, 49, 52, 69, 79, 80, 81]. However, much of this prior work did not target a clearly defined application, or focused on some simple toy problem without a clearly defined path to deal with messier real-world problems. Keeping the end application in mind is critical to building components which will be truly expandable to real-world problems and have a practical interface.

Admittedly, the implementation of feature detection in analog circuitry takes away the flexibility to modify the feature detector itself. However, since the Kanade feature detector is widely regarded as a robust front end, it is unlikely that one would need to modify it. Being a local computation, it is ideally suited for local implementation in an analog circuit. However, the aggregation of local data into a global estimate is better done by a single digital circuit to succeed the fabricated analog chip.

If a more compact system is desired, the path from an FPGA to a stand-alone digital chip (ASIC) is straightforward. The resulting system could be compact and have low power requirements, and thus be ideally suited for mobile robotics applications.

Chapter 2

Feature Detection for Motion Estimation

This chapter will focus on the algorithmic basis for a feature detector implemented in hardware, beginning with an overview of motion detection and culminating in a derivation of a feature detector appropriate for implementation in hardware. The highly repetitive nature of low-level image processing makes feature detection an ideal candidate for embedding in the circuitry of a custom chip, as discussed in chapter 1. However, a prior attempt at such an implementation [63] was debilitated by mismatch. To avoid this pitfall in my design, I modified the feature detector to allow incorporation of floating gates for mismatch reduction. This chapter includes simulations of the functionality of the implemented feature detector. Sec. 3.2 provides the implementation details of the circuits embodying my feature detector.

2.1 Motion Detection Overview

This section presents an overview of a motion estimation algorithm to provide background for understanding the remainder of this chapter. I explain how feature detection fits into the broader application of motion estimation and outline the basic steps of motion estimation, the first of which is feature detection and the focus of the current chapter.

As illustrated in fig. 2.1, the motion estimation algorithm can be subdivided into three steps. The second and third steps occur outside the photoreceptor array, in my system in an off-chip postprocessor, with the option of implementation on the same chip. The image and feature data become available only at discrete time intervals, the length of which is determined by how long it takes to read out the array. Each complete chunk of analog data about the entire array is termed a "frame".

1. **Detect features:** Within each frame, detect salient points in the image that will be easy to track from one frame to another.

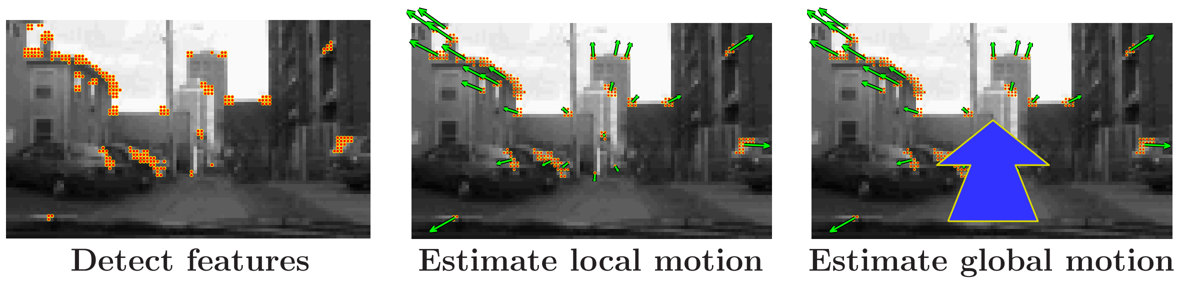


Figure 2.1: The motion estimation algorithm detects salient features within each image. It then estimates the local motion of each feature between successive frames. Based on these local motion vectors, it generates a global motion estimate.

2. **Estimate local motion:** For each detected feature, find the location to which it has moved in the next image frame. We assume a high frame rate relative to the speed of motion, so we need only search a local neighborhood of the feature to find its location one time step later. The vector pointing from the original feature location to the new location is the local motion vector for that feature.

3. **Estimate global motion:** Aggregate the local motion vectors to estimate a global motion vector that would have resulted in the local motions.

Good features for motion estimation are those that we can track easily over time, in this case from one frame to the next. To uniquely match an image patch in one frame to its corresponding patch in another frame, the feature which constitutes the image patch should be distinct from its neighborhood. Fig. 2.2 shows examples of image patches that make for poor features and good features to illustrate the idea. Areas of low contrast provide no trackable features. Panning a camera across a blank wall or other swath of uniform intensity results in little change to the neighborhood viewed by a given pixel, and thus little information about the motion. Low contrast variations can furthermore be overwhelmed by noise. High contrast edges are good indicators of motion perpendicular to the edge, but poor indicators of motion parallel to the edge. In the edge shown in the the center of fig. 2.2, each column of pixels views an almost identical image patch. Vertical motion would be undetectable. Thus, edges are poor general purpose features. The best features for tracking are those which can be distinguished from their neighborhoods regardless of the direction of image motion. A corner is an example of such a feature, as shown in the right of fig. 2.2 and in fig. 2.3. In general, a corner can be defined as an image patch having high intensity gradients along two axes.

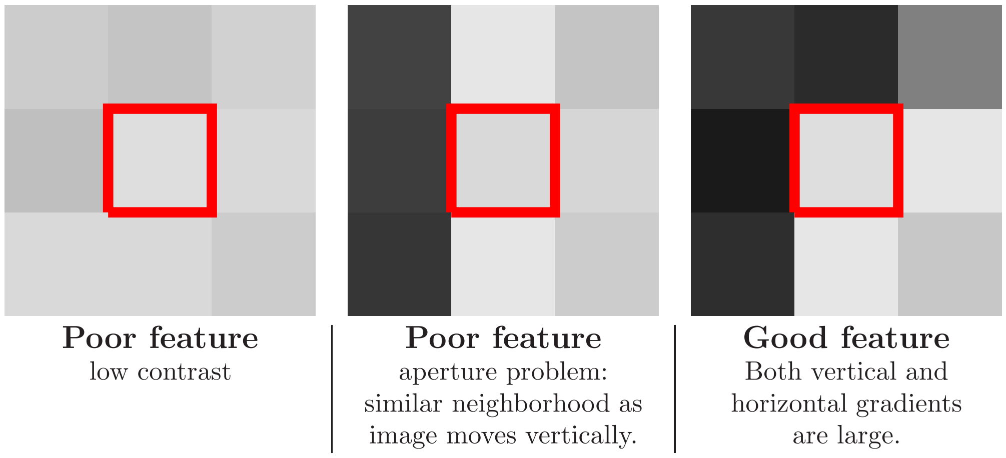


Figure 2.2: The images above show sample image patches that might be sensed by a 3x3 pixel grid. The central pixel (outlined in red) would use its neighborhood to determine a saliency value. The saliency value can be thresholded to decide whether the pixel is centered on a feature or not. Left: A patch of low contrast pixels. One low contrast patch is not uniquely different from a neighboring low contrast patch. Center: A high contrast edge. Motion parallel to the edge will result in little change to the central pixel's neighborhood. Thus, an edge is a poor feature for tracking motions having a component parallel to that edge. This is termed the "aperture problem" since in principle a larger viewing area would include elements that could provide correct information about parallel motion. Right: A feature that has a high intensity gradient in both the vertical and the horizontal direction can be tracked regardless of the direction of motion.

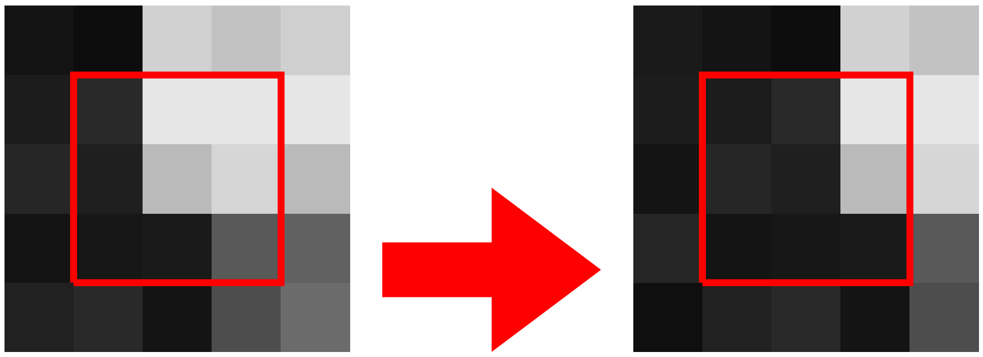


Figure 2.3: Local motion estimation: Consider the central 3×3 patch of pixels. This is the central pixel's immediate neighborhood. The motion of the corner in the image can be correctly calculated even from just this small local neighborhood.

2.2 Tomasi Kanade Features

Tomasi and Kanade [75] proposed a mathematically based description of a feature targeted at local motion estimation. Their definition fits the general overview presented in the previous section. The derivation begins by defining a feature as an image patch whose motion between two frames can be calculated, assuming translational motion with minimal other warping and assuming constant illumination. The resulting equations select as features those image patches having both large horizontal and large vertical intensity gradients, thus responding to corners and highly textured areas. They discard image patches having a large gradient along only one axis (edges) and low contrast areas. Tomasi-Kanade features have been used widely as the first step in algorithms for motion estimation.

Consider an image sequence $L(\vec{x}, t)$ where L denotes the light intensity, $\vec{x} = [x_h, x_v]^T$ denotes location within an image with x_h the horizontal coordinate and x_v the vertical coordinate, and t denotes the time when that image occurred within the sequence. We can express the motion between successive frames by motion vectors such that $\vec{d}(\vec{x}, t) = [d_h, d_v]^T$ expresses the motion at point $\vec{x}$ at time t, where d_h is the direction and magnitude of horizontal motion and d_v is the direction and magnitude of vertical motion. Assuming a high frame rate (temporal sampling frequency), we can assume that small image regions undergo a geometric transformation due to motion of the camera or within the image, but the intensities of objects remain the same. Given these assumptions, the transformation from one frame to next over a time interval τ can be expressed as:

$$L(\vec{x}, t) \quad = \quad L(\vec{x} + \vec{d}, t + \tau) \tag{2.1}$$

The task of the motion estimation algorithm is then to find $\vec{d}$ for each of a set of automatically selected point features in a pair of successive image frames in the sequence.

However, eqn. 2.1 will not be satisfied exactly. The assumption of intensity constancy is not perfect so the light intensity of an image patch L may change in a way not captured in eqn. 2.1, for example as objects move into a differently illuminated area. Points may also disappear or appear due to occlusion. Lastly, various sources of random noise exist, such as thermal noise within the sensor or distortion within the optics. Even the best estimate of $\vec{d}$ will not satisfy eqn. 2.1 perfectly. To estimate $\vec{d}$, then, find the $\vec{d}$ which minimizes the squared residual error:

$$\epsilon = \sum_{W} [L(\vec{x} + \vec{d}, t + \tau) - L(\vec{x}, t)]^2 \tag{2.2}$$

over a small image patch centered on $\vec{x}$ which we call the *feature window* W. Approximate $L(\vec{x} +$

$\vec{d}, t + \tau)$ by its Taylor series expansion:

$$\begin{aligned} L(\vec{x}+\vec{d}, t+\tau) &\approx L(\vec{x},t) + \left[\frac{\delta L(\vec{x},t)}{\delta x_h}, \frac{\delta L(\vec{x},t)}{\delta x_v}\right]\vec{d} + \frac{\delta L(\vec{x},t)}{\delta t}\cdot\tau \\ &= L(\vec{x},t) + \vec{G}(\vec{x},t)^T\vec{d} + L_t(\vec{x},t)\tau \end{aligned}$$

where $\vec{G}(\vec{x},t) = [G_h, G_v]^T$ denotes the horizontal and vertical intensity gradients at location $\vec{x}$ and time t, and $L_t(\vec{x},t)$ is the change in intensity over time. This expression incorporates the crucial assumption of this feature detection algorithm, namely, that the motion field can be locally approximated by a constant displacement of the local image patch.

We can then rewrite the residual error from eqn. 2.2 as:

$$\epsilon = \sum_W \left(\vec{G}(\vec{x},t)^T\vec{d} + L_t(\vec{x},t)\tau\right)^2$$

To minimize the residual, differentiate it with respect to $\vec{d}$, and set the result to zero:

$$\frac{\delta\epsilon}{\delta\vec{d}} = 2\cdot\sum_W \vec{G}(\vec{x},t)\cdot\left(\vec{G}(\vec{x},t)^T\vec{d} + L_t(\vec{x},t)\tau\right) = 0$$

Algebraic reshuffling of the terms and rewriting in matrix form leads to a definition of the *characteristic matrix* **C**.

$$\begin{aligned} \sum_W \vec{G}(\vec{x},t)\cdot\vec{G}(\vec{x},t)^T\vec{d} &= -\sum_W \vec{G}(\vec{x},t)\cdot L_t(\vec{x},t)\tau \\ \sum_W \begin{bmatrix} \left(\frac{\delta L(\vec{x},t)}{\delta x_h}\right)^2 & \frac{\delta L(\vec{x},t)}{\delta x_h}\cdot\frac{\delta L(\vec{x},t)}{\delta x_v} \\ \frac{\delta L(\vec{x},t)}{\delta x_h}\cdot\frac{\delta L(\vec{x},t)}{\delta x_v} & \left(\frac{\delta L(\vec{x},t)}{\delta x_v}\right)^2 \end{bmatrix}\cdot\vec{d} &= -\tau\sum_W \begin{bmatrix} \frac{\delta L(\vec{x},t)}{\delta x_h} \\ \frac{\delta L(\vec{x},t)}{\delta x_v} \end{bmatrix}\cdot\frac{\delta L(\vec{x},t)}{\delta t} \qquad (2.3) \\ \mathbf{C}\cdot\vec{d} &= \mathbf{e} \qquad (2.4) \end{aligned}$$

The displacement $\vec{d}$ can then be calculated as $\vec{d} = \mathbf{C}^{-1}\mathbf{e}$. The solution will be numerically stable only if the characteristic matrix **C** is well-conditioned and has entries well above the noise level. **C** will be well-conditioned if the eigenvalues of **C**, λ_1 and λ_2, do not differ by many orders of magnitude. The noise requirement is satisfied if λ_1 and λ_2 are both well above zero. In practice, the maximum magnitude of the eigenvalues is bounded by the maximum possible pixel value. Thus, the requirements can be simplified to:

$$min(\lambda_1, \lambda_2) > \lambda_t \qquad (2.5)$$

where λ_t is a threshold.

Features useful for motion estimation, then, are those image patches whose characteristic equation

C has eigenvalues exceeding some threshold λ_t, as expressed by the constraint in eqn. 2.5.

2.3 Prior aVLSI Implementation

To parallelize the computation of Tomasi-Kanade features, Pesavento [63, 61] built an analog VLSI (aVLSI) photoreceptor array with Kanade feature detectors integrated into the pixel array.

2.3.1 Overview of Pesavento's Chip

Each pixel contained modified Gilbert multipliers. As inputs, the multipliers took light intensity values detected by neighboring pixels. The circuits were wired such that the three multipliers output currents proportional to:

1. The square of the horizontal gradient, $\left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right)^2$
2. The square of the vertical gradient, $\left(\frac{\Delta L(\vec{x},t)}{\Delta x_v}\right)^2$
3. The product of the horizontal and vertical gradients, $\left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right) \cdot \left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right)$

To calculate the terms constituting the characteristic matrix, each of the above products needs to be summed over the several pixels in a feature window W. These summations are the components of the characteristic matrix:

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

1. $C_{1,1} = \sum_W \left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right)^2$
2. $C_{2,2} = \sum_W \left(\frac{\Delta L(\vec{x},t)}{\Delta x_v}\right)^2$
3. $C_{1,2} = C_{2,1} = \sum_W \left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right) \cdot \left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right)$

Since the multipliers output currents, summation can be done easily by simply connecting the outputs onto a single wire. The current flowing through this wire is then the sum of the incoming currents by Kirchhoff's Current Law. Indeed, the original version of Pesavento's design, *Detector1*, did exactly that. A 3x3 block of pixels constituted the feature window W. The multiplier outputs from the nine pixels were summed to produce $C_{1,1}$, $C_{2,2}$, and $C_{1,2}$.

The original Kanade algorithm calls for calculating the eigenvalues of the characteristic matrix and verifying that both are larger than some threshold λ_t. Pesavento derived a simplification of this calculation and showed that it was sufficient to instead threshold the quantity:

$$(C_{1,1}) \cdot (C_{2,2}) - (C_{1,2})^2 > p_t \tag{2.6}$$

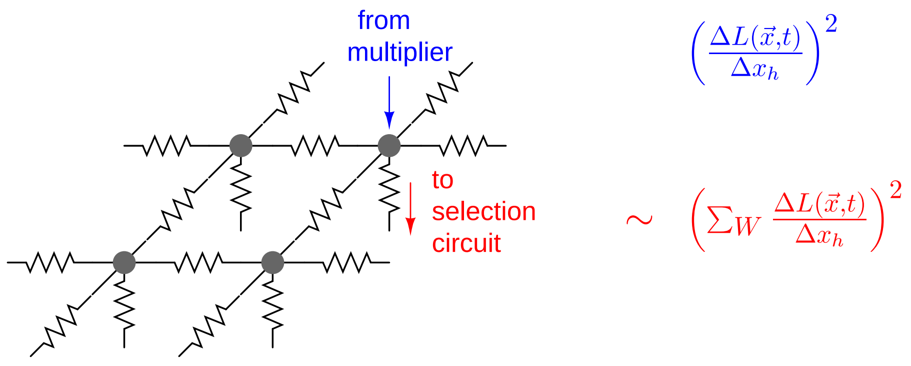

from multiplier
to selection circuit
$\left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right)^2$
$\sim \left(\sum_W \frac{\Delta L(\vec{x},t)}{\Delta x_h}\right)^2$

Figure 2.4: The resistive grid in Pesavento's *Detector2*, shown for $C_{1,1}$. Similar grids exist for $C_{1,2}$ and $C_{2,2}$. Each node represented by a dot corresponds to one pixel. Resistors connect neighboring pixels. The input to each node is the current output of a multiplier, in this case calculating the square of the horizontal gradient at that pixel. The output current flows through a resistor (drawn vertically) and to the selection circuit that determines whether or not a sufficiently salient feature exists that is centered on that pixel.

where p_t is the threshold [61]. The selection circuit which calculated the quantity in eqn. 2.6 was also implemented within each pixel.

Detector1 worked as expected. However, dedicating each 3x3 block of pixels to calculate only one feature wastes space, especially since with three modified Gilbert multipliers in each pixel the pixels are rather large. In an improved design, *Detector2*, Pesavento used a two-dimensional diffusion network (resistive grid) to allow overlapping use of multiplier outputs (see fig. 2.4). The resistive grid performs a weighted summation of nearby currents, approximating the sum over W. For example, the summation of $C_{1,1}$ at location $\vec{x} = [x_h, x_v]^T$ for a 3x3 pixel grid would be approximated as:

$$\sum_W C_{1,1}(x_h, x_v) \approx \begin{array}{lll} w_1 \cdot C_{1,1}(x_h+1, x_v-1) & + w_2 \cdot C_{1,1}(x_h, x_v-1) & + w_3 \cdot C_{1,1}(x_h-1, x_v-1) \\ + w_4 \cdot C_{1,1}(x_h+1, x_v) & + w_5 \cdot C_{1,1}(x_h, x_v) & + w_6 \cdot C_{1,1}(x_h-1, x_v) \\ + w_7 \cdot C_{1,1}(x_h+1, x_v+1) & + w_8 \cdot C_{1,1}(x_h, x_v+1) & + w_9 \cdot C_{1,1}(x_h-1, x_v+1) \end{array}$$

In the original Kanade formulation, $w_i = 1$ for all i. With a resistive grid, w_i corresponding to a nearby pixel has a larger value than one corresponding to a distant pixel. See fig 2.5 for a graphical representation.

The output currents of the resistive grids correspond to $C_{1,1}$, $C_{1,2}$, and $C_{2,2}$. In *Detector2*, these currents were input to a 47 transistor circuit which performed the thresholding of eqn. 2.6 and thus determined whether a feature was centered on the pixel or not.

The block diagram of *Detector2* is shown in fig. 2.6.

2.3.2 Mismatch Problems in Pesavento's Chip

While each feature detector of Pesavento's *Detector2* correctly identified features as prescribed by the Tomasi-Kanade algorithm, mismatch between circuits rendered the array unusable [62]. With

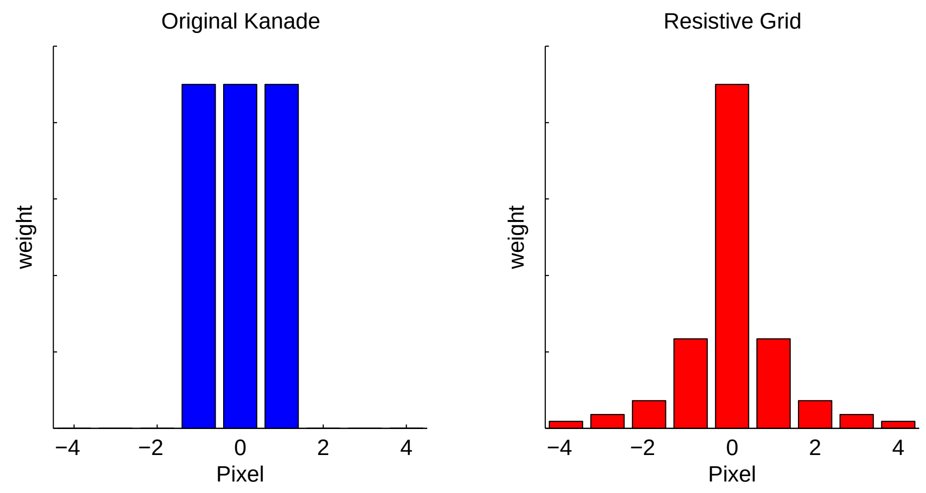


Figure 2.5: The resistive grid in Pesavento's *Detector2* changes the weighting of the summation of terms in the calculation of the characteristic matrix. Left: The pixels along a single row of the feature window W would be weighted uniformly and equally in the original Kanade algorithm. Pixels outside the feature window would be given a weight of zero, that is, not used in the calculation. Right: The resistive grid gives more weight to outputs from nearby pixels, and gradually decreasing weight to more distant pixels in the array.

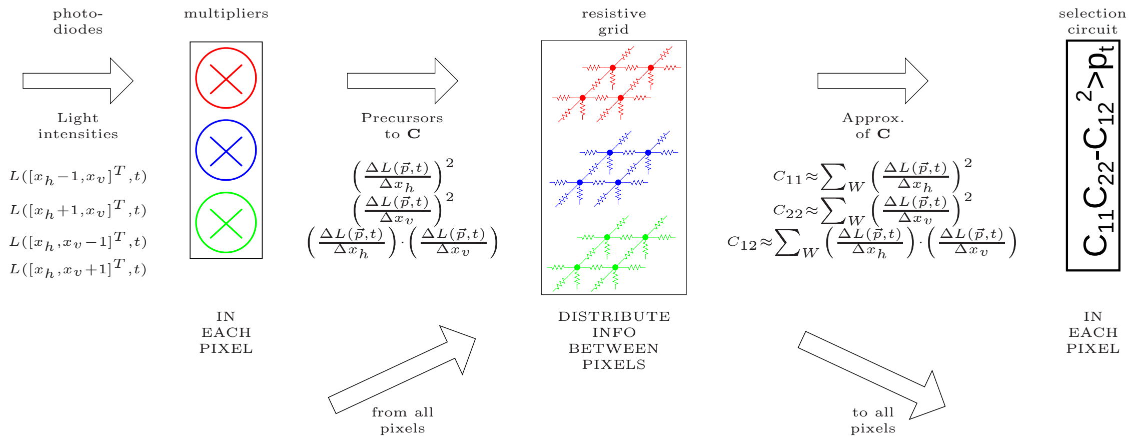


photo-diodes
Light intensities
$L([x_h-1,x_v]^T,t)$
$L([x_h+1,x_v]^T,t)$
$L([x_h,x_v-1]^T,t)$
$L([x_h,x_v+1]^T,t)$
multipliers
IN EACH PIXEL
Precursors to **C**
$\left(\frac{\Delta L(\vec{p},t)}{\Delta x_h}\right)^2$
$\left(\frac{\Delta L(\vec{p},t)}{\Delta x_v}\right)^2$
$\left(\frac{\Delta L(\vec{p},t)}{\Delta x_h}\right)\cdot\left(\frac{\Delta L(\vec{p},t)}{\Delta x_v}\right)$
from all pixels
resistive grid
DISTRIBUTE INFO BETWEEN PIXELS
Approx. of **C**
$C_{11}\approx\sum_W\left(\frac{\Delta L(\vec{p},t)}{\Delta x_h}\right)^2$
$C_{22}\approx\sum_W\left(\frac{\Delta L(\vec{p},t)}{\Delta x_v}\right)^2$
$C_{12}\approx\sum_W\left(\frac{\Delta L(\vec{p},t)}{\Delta x_h}\right)\cdot\left(\frac{\Delta L(\vec{p},t)}{\Delta x_v}\right)$
to all pixels
selection circuit
$C_{11}C_{22}-C_{12}{}^2>p_t$
IN EACH PIXEL

Figure 2.6: Diagram of *Detector2*. Each pixels shares its detected light intensity value with its neighbors. Each pixel takes the difference of these intensities (the gradient) and multiplies them to generate precursors of the terms in the characteristic equation **C**. A sum of these values is approximated by resistive grids shared among all pixels. Each pixel then draws values from the resistive grids as inputs to its selection circuit. The selection circuit within each pixel produces a binary value to indicate whether a feature is centered on that pixel.

all voltage bias settings configured as best possible, some pixels always indicated a feature, while others never flagged the presence of a feature even when surrounded by a highly salient image patch. Neither the multipliers nor other circuits within the pixel incorporated techniques for mismatch minimization.

Pesavento suspected that the mismatch was primarily due to mismatch within the three multipliers used to calculate the characteristic matrix. This is a plausible explanation. Small differences in threshold voltages could lead to significant errors when amplified by a multiplication operation.

My simulations of mismatch within a resistive grid suggest approximately 17% variation in the magnitude of the currents representing $C_{1,1}$, $C_{1,2}$, and $C_{2,2}$ when minimum size transistors are used, 8.5% with both length and width doubled (4x area), 4.4% with both length and width quadrupled (16x area). Thus, mismatch within the resistive grid would have impaired performance but should not have rendered the array useless.

Mismatch in the selection circuit has not been quantified by either simulations or measurements. However, this circuit may contribute as much to the mismatch problems as the multipliers. Since it relies on current mirrors for subtraction, small differences in threshold voltages would be reflected in the magnitudes of the currents. These currents are then multiplied, an operation performed by using subthreshold transistors cleverly. Mismatches in the voltage thresholds of these transistors would contribute further to mismatch errors. In short, the sources of mismatch in the modified Gilbert multipliers and mismatch in the selection circuit are similar. Further analysis would be required to tease apart the exact contribution of each.

2.3.3 Brief Overview of Floating Gates for Mismatch Reduction

We chose to use floating gate devices to reduce mismatch in these circuits, as explained in detail in sec. 3.1.2. Layout techniques are inappropriate or insufficient. Common-centroid layout can be used to reduce mismatch between nearby transistors but is not applicable to hundreds of transistors nor transistors on opposite ends of a large array. Photolithographically invariant layout would not improve matching sufficiently. Simply making larger devices requires too much area when we desire an array of identical circuits. Floating gate devices can be programmed permanently to modify the offset of a single transistor's operating point so as to null the mismatch introduced by imperfections within the single circuit within which the transistor is embedded.

Floating gate devices are described in detail in sec. 3.1.4. In brief, a floating gate transistor is a transistor whose gate is fully isolated. Charge stored upon the gate sets the gate voltage. The amount of charge can be modified by tunneling or hot-carrier injection, processes that do not occur under normal circuit operation. The charge is stored in a non-volatile manner, with retention times on the order of years or decades. Thus, we are able to program the floating gates to remove offset initially and operate the chip without further reprogramming.

2.3.4 Reducing Circuit Mismatch in Non-linear Circuits

Any technique for mismatch correction should be applied judiciously, since some cost or performance trade-off is typically required. Floating gates require some additional layout area and increased operational complexity to initialize their values appropriately prior to operation. One way to consider the number of floating gates appropriate for a given circuit is to analyze the number of degrees of freedom that must be programmed to compensate for the major sources of mismatch.

Given a series of linear stages, only a single floating gate is necessary since the mismatch can be expressed as a single term requiring a correction with only one degree of freedom. Consider a series of circuits with linear transfer functions $H_i(V) = A_i * V$, each introducing some mismatch V_i, where i is an index referring to each circuit. For an input V_{in}, the output of the series will then equal V_{out}, as expressed below.

$$\begin{array}{llllllll} \hat{V}_{out} = & A_n * (& A_{n-1} * (& A_{n-2} * (& ... & * & A_1 * V_{in}) & ... \quad))) \quad \texttt{ideal} \\ V_{out} = V_n + & A_n * (V_{n-1} + & A_{n-1} * (V_{n-2} + & A_{n-2} * (& ... & *(V_1 + & A_1 * V_{in}) & ... \quad))) \quad \texttt{mismatched} \end{array}$$

Denote ideal output assuming no mismatch by $\hat{V}_{out}$ and output including mismatch by V_{out}. Simplify and combine the two equations:

$$\begin{aligned} V_{out} &= V_n + (A_n * V_{n-1}) + (A_{n-1} * V_{n-2}) + ... + (A_2 * V_1) + \hat{V}_{out} \\ &= \hat{V}_{out} + \underbrace{\sum_{n=1}^{N-1} A_{n+1} * V_n}_{mismatch\ term} \end{aligned}$$

Since we are considering linear circuits, the gain terms, A_i, are scalars and thus the mismatch term is a constant. Compensating for the mismatch requires adjusting the value of a single term.

For every added degree of freedom in the circuits, we need to add one more floating gate to compensate. For example, a multiplier has two degrees of freedom which can be expressed as gain and offset:

$$V_{out} = A_{gain} * (V_{in1} * V_{in2}) + V_{offset}$$

or equivalently, as offsets within each input term:

$$V_{out} = (V_{in1} + V_{offset1}) * (V_{in2} + V_{offset2})$$

Regardless of how we reshuffle the terms, it is impossible to generate an expression where the mismatch will be encapsulated into a single term. Thus, we must correct for the mismatch along two degrees of freedom. In this case, we could consider using two floating gates, one which corrects for V_{offset} and a second which corrects for A_{gain}. Alternately, the roles of the floating gates could be to correct for $V_{offset1}$ and $V_{offset2}$, respectively. The choice should rest on ease of implementation, that is, which expression can be better translated into a reasonable circuit topology.

Lastly, note that the mismatch in one circuit sometimes can be compensated in a succeeding circuit without introducing modifications beyond what already exists in that latter circuit. For example, a difference circuit has one degree of freedom, requiring one floating gate to correct. A multiplier has two degrees of freedom, requiring two floating gates. A multiplication of a difference, implemented as a difference circuit followed by a multiplier, has only two degrees of freedom. The mismatch in this combined circuit can then be compensated for by the two floating gates in the multiplier, making a floating gate in the difference circuit extraneous. The outputs of these two circuits and of their combination are expressed in the table below. Note that the form of the expression for the mismatch in the combined circuit is the same as that of the multiplier alone. Offset mismatches are represented by V_{os1}, V_{os2}, and V_{os3}. Gain mismatches are denoted by A_{m1} and A_{m2}. The inputs to each circuit are expressed as V_{i1} and V_{i2}.

circuit:	**expression**
difference:	$V_{diff} = V_{i1} - V_{i2} + V_{os3}$
multiplier:	$V_{mult} = (V_{i1} + V_{os1}) * (V_{i2} + V_{os2})$
both:	$V_{out} = (V_{diff1} + V_{os1}) * (V_{diff2} + V_{os2})$
	$= (V_{i1} - V_{i2} + \underbrace{V_{os3} + V_{os1}}_{mismatch\ term}) * (V_1 - V_2 + \underbrace{V_{os3} + V_{os2}}_{mismatch\ term})$

In contrast, consider a different example where the mismatch terms for one circuit are only partly absorbed by the successive circuit. A squaring operation and exponentiation each have two degrees of freedom. The square of an exponent has three degrees of freedom.

circuit:	**expression**
square:	$V_{sqr} = A_{m1} * (V_{i1} + V_{os1})^2$
exponential:	$V_{exp} = V_{os2} + A_{m2} * e^{V_{i1}}$
both:	$V_{out} = A_{m1} * (V_{exp} + V_{os1})^2$
	$= \underbrace{A_{m1}} * (\underbrace{A_{m2}} * e^{V_{i1}} + \underbrace{V_{os1} + V_{os2}}_{mismatch\ term})^2$

A general rule to determine whether a successive circuit can be used to compensate for mismatch in a previous circuit does exist. If the mismatch in one circuit can be expressed in the same form as one of the mismatch terms in a succeeding circuit, then the terms can be merged. In other words, the number of floating gates necessary to remove all offsets is equal to the number of degrees of freedom in a circuit.

2.3.5 Requirements for Mismatch Correction in Pesavento's Circuits

Consider the number of degrees of freedom in Pesavento's feature detector to assess how many floating gates would be necessary to correct for mismatch within the constituent circuits.

The first stage of processing consists of the modified Gilbert multipliers which calculate a difference ($\frac{\Delta L}{\Delta x_h}$ or $\frac{\Delta L}{\Delta x_v}$, the horizontal or vertical light intensity gradients) and a product of these differences. This stage can exhibit mismatch along two degrees of freedom. Both from Pesavento's experience and from an understanding of the circuits and the nature of the multiplication operation, we can know that even minor mismatch in the threshold voltage of the transistors can significantly impair the matching of the multiplier outputs. Mismatch compensation is necessary.

The next stage consists of the resistive grid, which calculates a weighted sum of the multiplier outputs. Summation is a linear operation. Since the output of each multiplier is split between several outputs of the resistive grid, mismatch within the resistive grid cannot be corrected in a straightforward manner within the multipliers. A useful correction in one multiplier for one output could lead to an exactly wrong correction for another output. However, the magnitude of the mismatch within this circuit is small enough that slightly increasing the size of the constituent transistors could suffice. Further simulations would need to be done.

The last stage consists of the calculation of $(C_{1,1}) \cdot (C_{2,2}) - (C_{1,2})^2 > p_t$ from eqn. 2.6. We can note two degrees of freedom in the multiplication, and another two in the squared term. A total of four floating gates would be required to remove mismatch in this stage.

In total, ten floating gates would be necessary to compensate for mismatch. Six floating gates would compensate for the two degrees of freedom in each of the three Gilbert multipliers. Four floating gates would compensate for mismatch in the selection circuit. This is an unreasonable number of floating gates. First, the layout area needed for all ten of the floating gates would enlarge the pixel significantly. The primary layout cost comes from the increased spacing needed between the wells for the tunneling junctions and wells in the rest of the circuits. Secondly, programming ten floating gates could require a complex procedure. Programming is straightforward if a single variable can be observed to gauge the progress of the programming procedure. With ten degrees of freedom, intermediate pixel outputs would need to be routed for readout, possibly requiring amplification, certainly requiring switching onto shared buses. While possible, this solution is unwieldy.

2.4 Orthogonal Gradient Detector (OGD)

I simplified the feature detector to a level where the necessary mismatch correction could be reasonably implemented. A straightforward application of mismatch reduction techniques to the topology fabricated by Pesavento seemed to require too many undesirable trade-offs.

2.4.1 Reducing the Kanade Detector

Let us consider the intuitive meaning of the equations describing the Kanade feature detector (eqns. 2.3–2.5). The Kanade feature detector selects as features those image patches whose characteristic matrix has large eigenvalues. This matrix is rewritten below using slightly simpler notation than previously for clarity. Let G_v be a vertical gradient at a single pixel and G_h the horizontal gradient. Use $\sum G_v^2$ to represent the sum of the squares of all vertical gradients within a window, and analogously $\sum G_h^2$ and $\sum G_h G_v$ to represent the other terms of the matrix. Then:

$$C = \begin{bmatrix} \sum G_h^2 & \sum G_h G_v \\ \sum G_h G_v & \sum G_v^2 \end{bmatrix}$$

To find the eigenvalues, we set the determinant of the characteristic matrix to zero:

$$\begin{aligned} \begin{vmatrix} (\sum G_h^2) - \lambda & \sum G_h G_v \\ \sum G_h G_v & (\sum G_v^2) - \lambda \end{vmatrix} &= 0 \\ \left(\left(\sum G_h^2\right) - \lambda\right) \cdot \left(\left(\sum G_v^2\right) - \lambda\right) - \left(\sum G_h G_v\right) \cdot \left(\sum G_h G_v\right) &= 0 \\ \underbrace{\sum G_h^2 \sum G_v^2 - \left(\sum G_h G_v\right)^2}_{c} + \underbrace{\left(-\sum G_v^2 - \sum G_h^2\right)}_{b} \lambda + \underbrace{1}_{a} \lambda^2 &= 0 \end{aligned} \tag{2.7}$$

To solve for the eigenvalues λ, we can use the quadratic equation that reduces for $a = 1$ to:

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad \rightarrow \quad 2\lambda = -b \pm \sqrt{b^2 - 4c}$$

For both λ to be large, b needs to be large and the square root term needs to be small. For b to be large, either one or both gradients (G_h and G_v) need to be large at many pixels within the feature window. Minimizing the square root term requires b^2 and $4c$ to be similar in magnitude, which will be the case if G_h and G_v are of similar magnitude at many points within the feature window. Both these conditions are true if many pixels observe both a large vertical and a large horizontal gradient. Pixels detecting significantly different gradients along the two axes will adversely affect the term $(b^2 - 4c)$, while pixels detecting gradients of small magnitude will adversely affect b.

Scaling the Kanade detector to fewer pixels, consider the most minimal case of a feature window consisting of only one pixel calculating a single horizontal gradient and a single vertical gradient. The calculation of the eigenvalues can be written directly, after eqn. 2.7:

$$\begin{aligned} G_h^2 G_v^2 - (G_h G_v)^2 + \left(-G_v^2 - G_h^2\right) \lambda + \lambda^2 &= 0 \\ G_v^2 + G_h^2 &= \lambda \end{aligned}$$

This mathematically correct and stable answer is nonetheless unsuitable in that it fails to enforce the critical constraint that *both* gradients be large.

2.4.2 Definition of the Orthogonal Gradient Detector (OGD)

Consider, then, what would constitute a minimal unit that embodies some of the basic objectives of the Kanade feature detector. At its smallest, it must calculate a single horizontal gradient, a single vertical gradient, and a binary function indicating whether both gradients were above a threshold. A mathematical description can be represented by either of the equivalent statements:

$$(abs(G_h) > thold) \quad \text{and} \quad (abs(G_v) > thold)$$
$$min(abs(G_h), abs(G_v)) > thold$$

or using notation from earlier in the section:

$$abs\left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right) > thold \quad \text{and} \quad abs\left(\frac{\Delta L(\vec{x},t)}{\Delta x_v}\right) > thold$$
$$min\left(abs\left(\frac{\Delta L(\vec{x},t)}{\Delta x_h}\right), abs\left(\frac{\Delta L(\vec{x},t)}{\Delta x_v}\right)\right) > thold$$

Call this the Orthogonal Gradient Detector (OGD).

The difference between a Kanade detector and the OGD is twofold. First, the Kanade detector can be straightforwardly applied to any size feature window, obviously requiring more computation in proportion to the number of pixels evaluated, while the OGD by definition considers exactly four neighboring pixels. Second, the two detectors differ in the ordering of the two steps of accumulation of information over the feature window and of thresholding. The Kanade detector first accumulates the products of gradients into the characteristic matrix, and secondly applies a threshold to the eigenvalues of that matrix. The OGD applies the threshold to each pixel. If multiple OGDs were to be aggregated, that accumulation would be done as a second step. The Kanade detector enforces a structural constraint on the local image patch in the manner in which it aggregates information from the constituent pixels. The OGD does not have an inherent mechanism for responding to structure on a larger scale.

2.4.3 Comparison of Kanade detector and OGD

The Kanade detector and the OGD differ in the types of features they detect. Most obviously, the Kanade detector uses a larger patch of the image and thus has more information from which to decide whether a given image patch is salient. For the Kanade detector, the calculation of the eigenvalues of the characteristic matrix (see eqn. 2.3) is trivial and meaningless if only the gradients for a single pixel are used. Minimally, gradients from several pixels are necessary. For the feature calculation to be centered conveniently on a pixel rather than a location between pixels, a 5x5 pixel window provides a minimal 3x3 patch of gradient values. In comparison, the OGD relies on only a single vertical gradient and a single horizontal gradient, thus incorporating information from 4 pixels. Secondly, since the Kanade detector responds to structure within this larger pixel window, it responds to more complex structure than the OGD. Using G_h and G_v to represent the horizontal and vertical gradients, respectively, at each pixel within the feature window W, rewrite eqn. 2.3 for the characteristic matrix C:

$$\underbrace{\sum_W \begin{bmatrix} G_h^2 & G_hG_v \\ G_hG_v & G_v^2 \end{bmatrix}}_{\mathbf{C}} \cdot \vec{d} = -\tau \begin{bmatrix} G_h \\ G_v \end{bmatrix} \cdot L_t \tag{2.8}$$

The eigenvalues are then the solution to the quadratic equation:

$$\underbrace{\sum G_h^2 \sum G_v^2 - \left(\sum G_hG_v\right)^2}_{c} \lambda + \underbrace{\left(-\sum G_v^2 - \sum G_h^2\right)}_{b} + \underbrace{1}_{a} \lambda^2 = 0$$

$$2\lambda = -b \pm \sqrt{b^2 - 4c}$$

A feature is defined where both eigenvalues are large and of a similar order of magnitude. In practice, the latter constraint is enforced by limitations on the maximum possible pixel values, and only the former constraint is of concern. The b term enforces response to high contrasts. The c term enforces response to patches having structure, such as corners or textures, while reducing response to linear edges. The OGD has no analogous means of enforcing constraints on structure.

In practice, the features selected by both detectors in response to the image in fig. 2.7 are shown in fig. 2.8. The 30 best features selected by each feature detector are shown. The red outlines indicate which pixels within each image patch were used for the feature calculation. Both detectors respond positively to corner-like features. The Kanade detector also responds to end-points of lines, which are composed of two corners. As a result of its inability to consider structure over a larger window, the OGD detector responds to diagonal lines as it would to a sequence of corners, while the Kanade detector correctly rejects them as it would any other edge.

From the perspective of hardware implementation, the two detectors differ in that the mathematical computations involved in the OGD are reduced relative to the Kanade detector in a manner that facilitates implementation in hardware. Pesavento's Kanade detector requires 3 multiplications to calculate the characteristic matrix, plus another two multiplications and one subtraction to calculate the featureness value, plus a thresholding operation. The OGD requires two subtractions to calculate the gradients, two thresholding operations, and a single AND. The OGD can be implemented with simpler circuits and ones whose mismatch can be corrected with far fewer floating gates, since

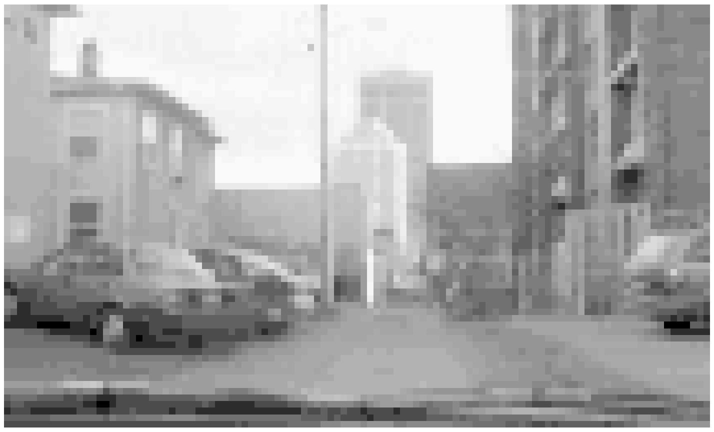


Figure 2.7: A sample image used to generate the features shown in fig. 2.8

the circuits have fewer degrees of freedom. The details of the circuit implementation of the OGD are described in sec. 3.2.

To complete the comparison of the Kanade detector and the OGD, I considered how well the detected features could be applied to local motion estimation. For every feature in an image frame, I compared the 3×3 pixel patch surrounding that feature with 3×3 patches surrounding nearby features in the succeeding frame to calculate the local motion at a feature. Local motion is calculated to be that which would take the feature in the first frame to its best matched nearby patch in the succeeding frame. I limited the search to neighboring image patches, rather than searching the entire image. This limit on search area reduces the computational requirements, improves matching for small motions by effectively rejecting matches to distant image locations, and restricts the magnitude of detectable motions. The limit must thus be chosen with regard for the fastest motion that the detector may need to detect. The matching was not further optimized for the detector comparison presented herein. For example, if feature A matches A', then B is nonetheless also allowed to match A'. This simplistic matching algorithm is illustrated in fig. 2.9. In hardware, such comparisons could be performed using a modified bump-circuit [42, 19] incorporating floating gates to remove mismatch, with the output currents simply summed onto a wire to calculate an overall similarity measure of each patch. The comparisons could be done serially for each feature within the image. Since the number of useful features within the image is presumably much smaller than the number of pixels within the image, the additional time necessary to perform this calculation would be reasonable.

Using this method of computing local motion, I compared the fraction of features at which the resulting local motion estimate was correct for the two feature detectors. I considered a number of image sequences, most outdoors. For a fair comparison, the thresholds for the detectors were set such

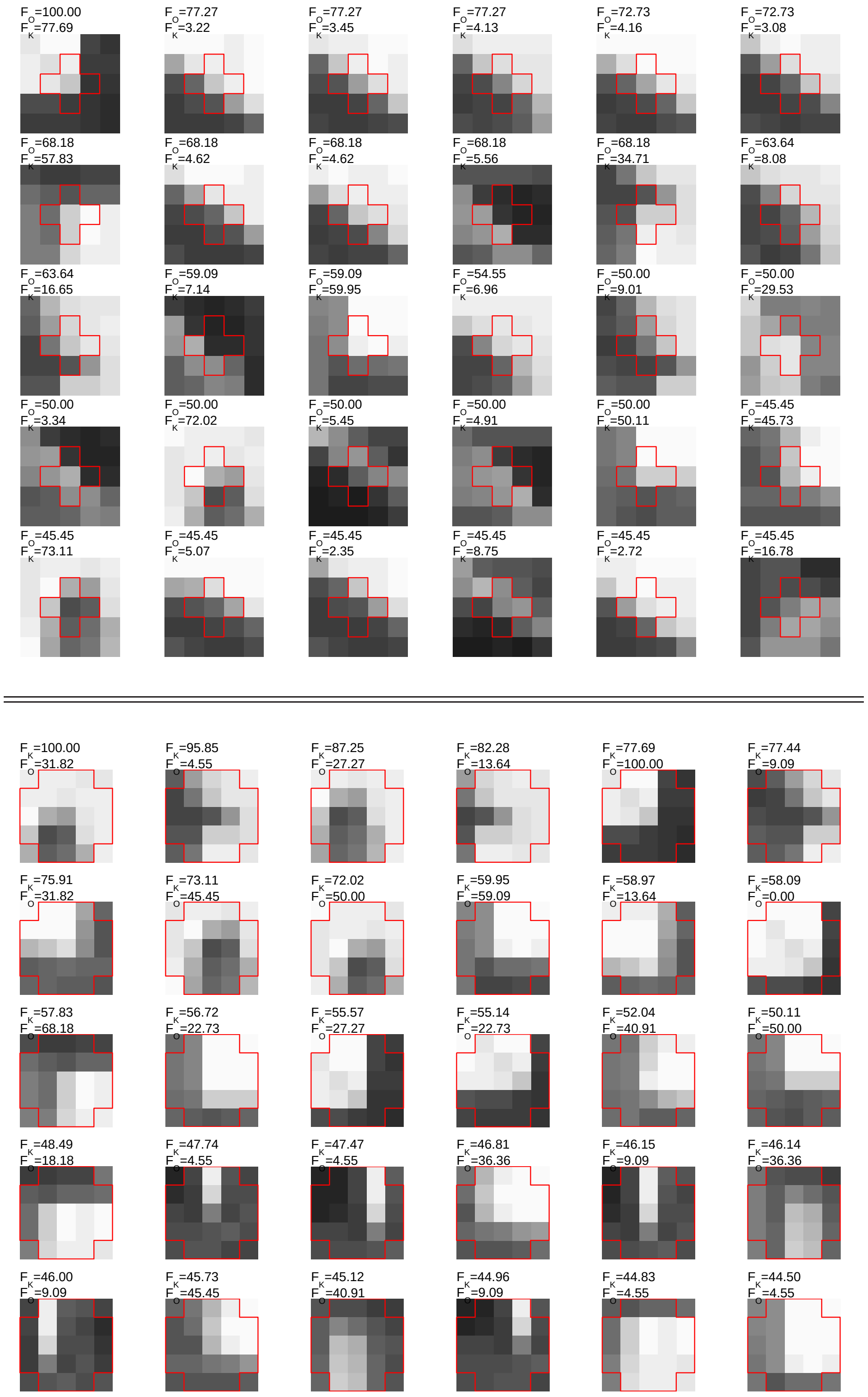


Figure 2.8: The 30 best features detected by a 3x3 Kanade feature detector (bottom) and the Orthogonal Gradient Detector (top) in the image from fig. 2.7. The red outline indicates pixels considered by each detector. Values above each patch are the "featureness" of that image patch, that is, the value that is thresholded to determine whether to classify the patch as a feature. For the 3x3 Kanade detector, F_K is the smaller of the eigenvalues λ. For the OGD, F_O is the smaller of the gradient values. Both F_K and F_O are normalized to range between 0 and 100 within the image. Implications of differences between the detectors are discussed in the text and in figs. 2.10–2.16.

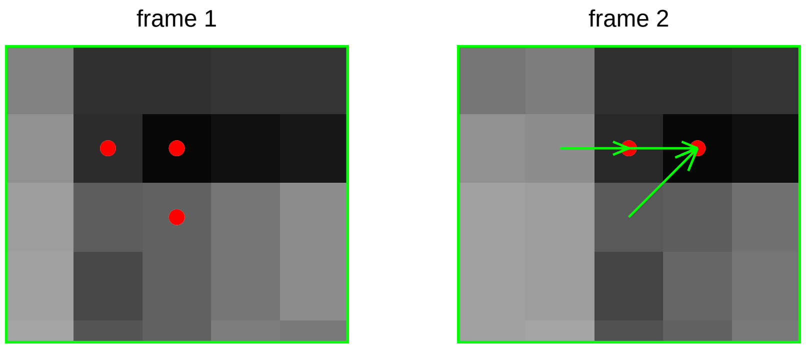


Figure 2.9: Example of detecting local motion. To find the local motion, the 3×3 image patch around each feature in frame 1 is compared to equally-sized patches around neighboring images in frame 2. The patch bearing the most similarity is assumed to be at the destination location of that feature. Red dots indicate features detected in this small patch of an image. The green arrows in frame 2 indicate the calculated motions between frames 1 and 2. Note that since one of the features was lost in the second frame, the motion of one of the three features is estimated erroneously.

that the numbers of features detected in the first frames of each sequence were almost the same for each detector and each image sequence. I used motion sequences with small displacements such that the motion between frames was approximately equal to the spacing between pixels. Thus, a correct local motion estimate would indicate motion to one of the neighboring eight pixels or no motion, and I could aggregate the local motion estimates into a fraction of correctly-estimated motions. The results are shown in figs. 2.10–2.16. The Kanade detector consistently performs slightly better, but the margin of improvement varies with the type of image and is often very slim. As mentioned previously, the OGD is sometimes confounded by diagonal edges. In highly structured scenes with many non-textured areas, such as parking lots and other city scenes, both detectors perform well with only a small margin of improvement for the Kanade detector. Such scenes simply do not present the sorts of complex features where the added benefits of the Kanade detector would yield a significant improvement. Some textured images, however, such as the chairs in fig. 2.15, contain repetitive structure that provides the Kanade detector with good features but confuses the OGD as it responds to the many lines and unstructured high-contrast areas. Perhaps surprisingly, in spite of its simplicity, the OGD sometimes outperforms the more sophisticated Kanade detector (ex. fig. 2.16). While the Kanade detector looks for trackable local structure, it does not verify that neighboring image locations do not have very similar appearances. Thus, sometimes the OGD selects features which are trackable by virtue of having high contrast and being different from their surround, such as edges flanked by lower contrast patches. The OGD's results tend to be more uniformly distributed, while the Kanade detector's features tend to clump in a smaller number of locations within the image. In some sense, then, the OGD has better coverage since the features provide information about more areas of the image than the Kanade detector.

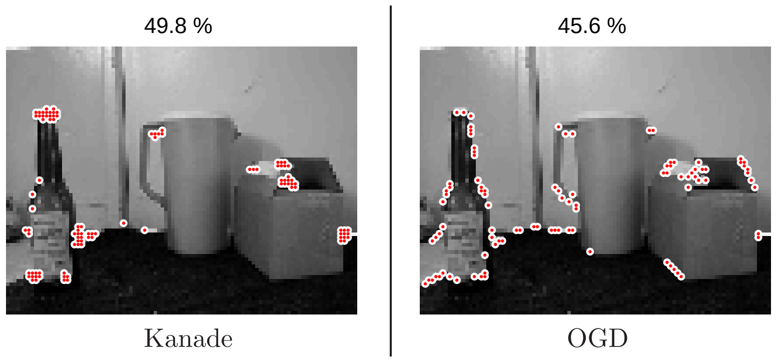


Figure 2.10: This synthetic image shows some of the fundamental differences between the Kanade detector and the OGD. Each red dot marks one feature. The percentages listed above the images indicate the fraction of features for which local motion was correctly estimated, as described in the text. Note that the Kanade detector responds strongly to high-contrast areas of local structure. The OGD selects both those areas and also several edges, which it confuses for a series of salient features. However, enough variation exists at those edges for the percentage of correctly tracked features to be only slightly lower for the OGD.

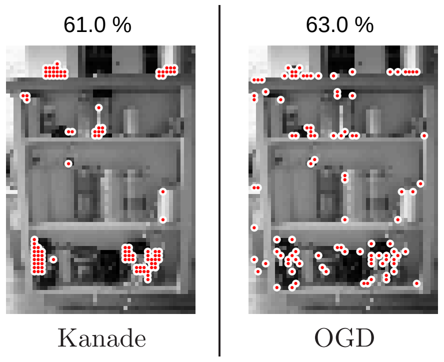


Figure 2.11: In images with much structure and thus easily-trackable points, both the Kanade and the OGD perform similarly. There are few diagonal lines and highly textured areas that would confuse the OGD in a manner manageable by the Kanade detector.

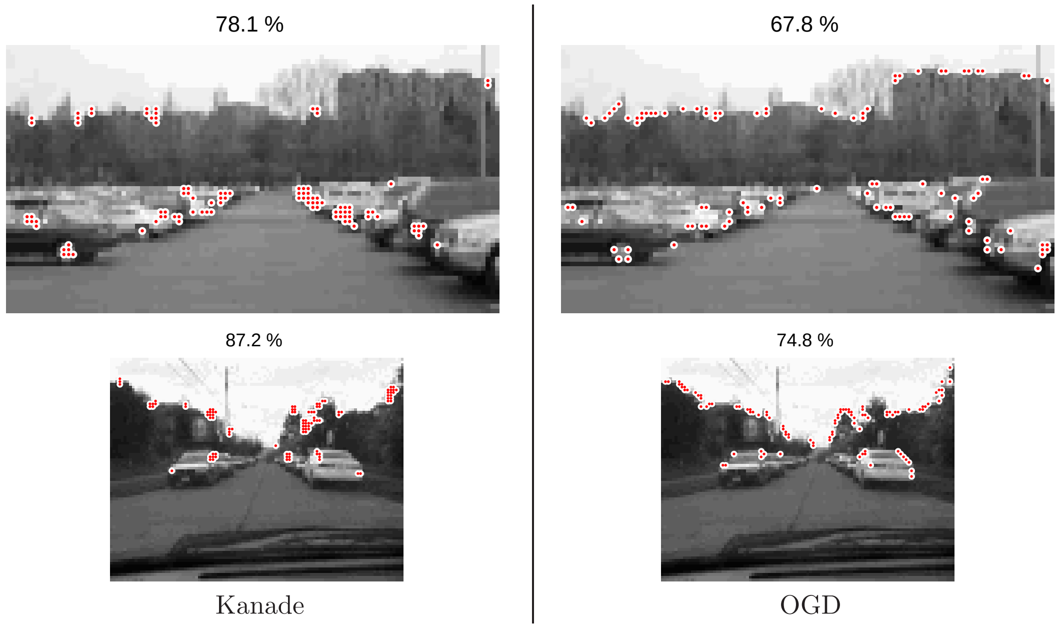


Figure 2.12: Many city scenes contain simple structures, such as cars, and the performance difference between the two detectors seen here is typical.

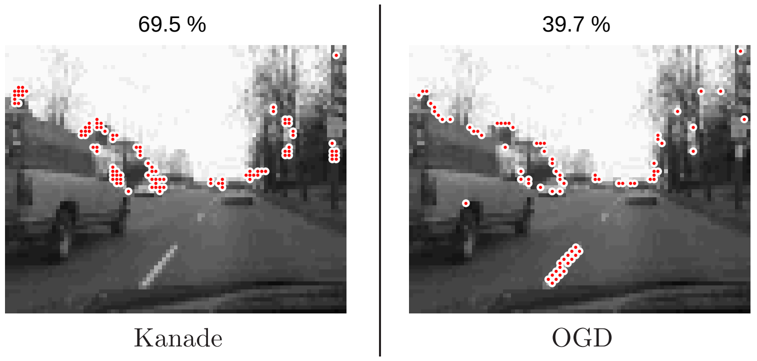


Figure 2.13: In this image, the diagonal lane marker vastly reduces the effectiveness of the OGD. Because the OGD responds to diagonal lines, the chosen “features” along the edges of the line provide poor locations for motion estimation and thus incorrect local motion estimates.

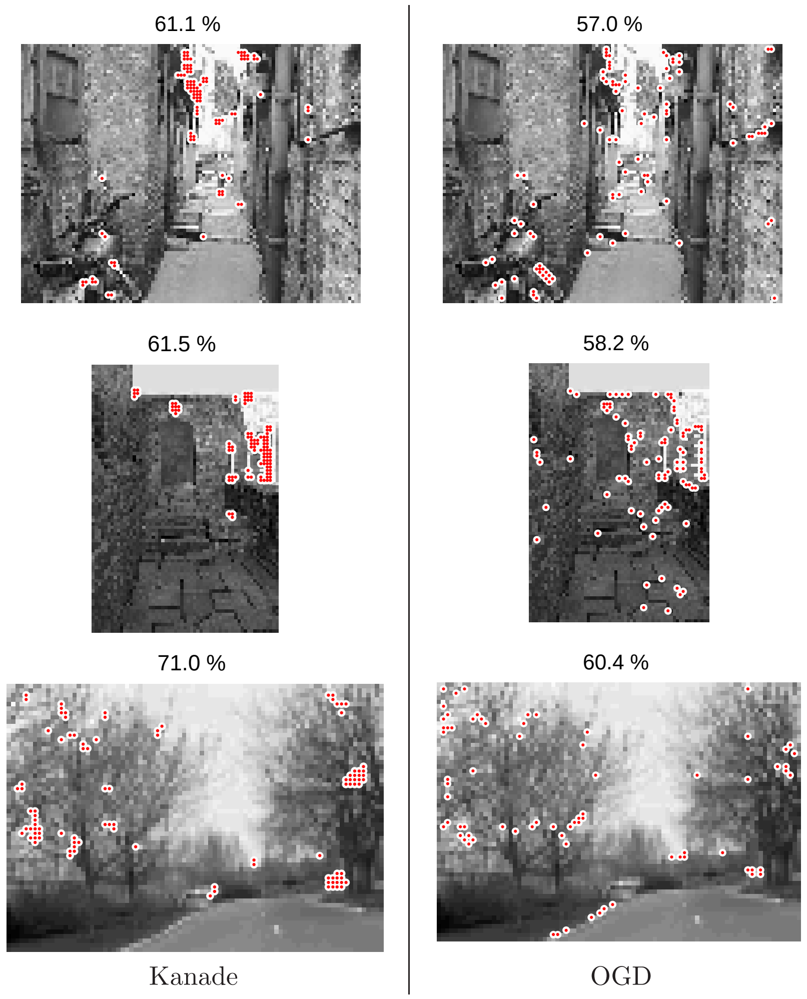


Figure 2.14: Many highly textured images result in local motion estimates of similar quality, surprisingly. While one might expect that the ability of the Kanade detector to detect structure would be especially critical in such images, in practice it turns out that local structure provide a small advantage over simply finding very high-contrast spots within the image.

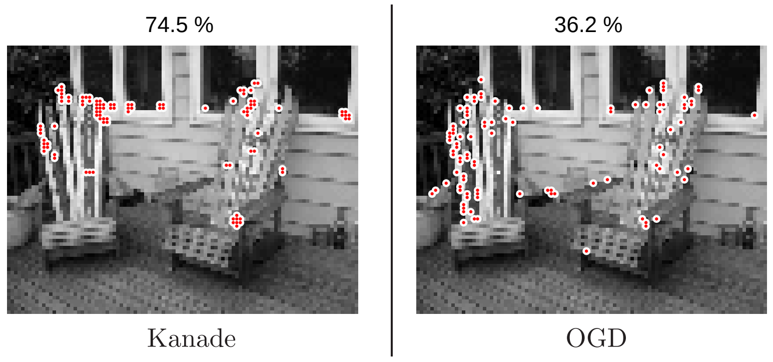


Figure 2.15: In images with relatively high frequency structure, the Kanade detector significantly outperforms the OGD. The backs of the chairs have spatial structure that is picked out well by the Kanade detector. For the OGD, on the other hand, each chair is simply a highly textured area with one spot no more salient than another. It is unable to consistently recognize the same features.

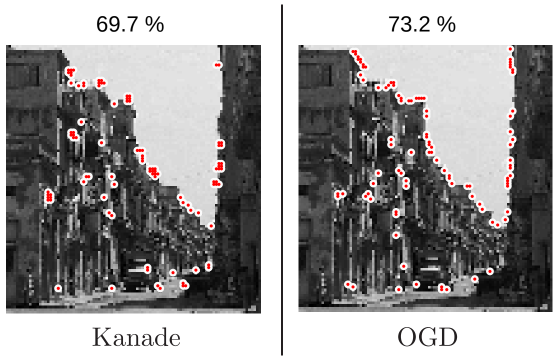


Figure 2.16: An unusual example wherein the OGD performs better than the Kanade detector. The jagged skyline with its diagonal lines is preferred by the OGD. Enough variation exists in the texture of the buildings that the features detected along the diagonal edges suffice for good local motion estimation.

Chapter 3

Circuit Design and Chip Data

3.1 Mismatch Reduction

Mismatch is a critical design issue in many circuits. Imperfections in chip manufacturing result in circuit elements which are not quite identical and thus function differently than designed. People have addressed the topic from many sides. Advancements in modeling accuracy assist in understanding the repercussions of mismatch on a given circuit and facilitate choosing appropriate transistor sizes. Improvements in processing address the fundamental physical reasons for the mismatch. Layout techniques are available to the circuit designer to improve matching within the boundaries of a given fabrication process. Other circuit design techniques focus on designing circuits that are less sensitive to process variations or that can be compensated in some manner after fabrication.

This section reviews the impacts of mismatch as applicable to our circuits. Many layout techniques rely on larger transistor sizing, which is impractical in arrays since each structure is repeated multiple times. We instead chose to use floating-gate transistors, initialized once to null mismatch. This section reviews both layout techniques and the use of floating-gate transistors for mismatch reduction.

3.1.1 Briefly, on Modeling

Most approaches to modeling mismatch are based on the seminal work by Pelgrom et al. [60], including the commonly used SPICE simulation parameters. Michael and Ismail extended Pelgrom's work to make the extracted parameters more compatible with existing circuit simulators and in [55] thoroughly reference work prior to their own, providing a good review of the state of the art at the time. While the Pelgrom parameters are computationally useful for modeling and simulating mismatch, they do not directly map onto specific physical parameters. This makes them less useful for understanding how to adjust the fabrication process to reduce mismatch, a concern addressed by Drennan in [26]. Additional literature on mismatch modeling is profuse and research on this

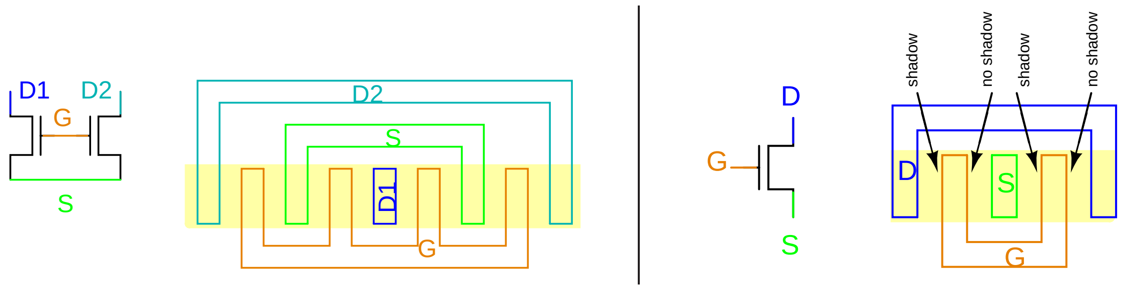


Figure 3.1: Layout for mismatch reduction. Left: Common-centroid layout improves matching between nearby transistors. Consider what happens if a linear doping gradient exists on the chip such that the left side of the figure receives a lower dopant concentration. The transistor formed by the left side of D2 and S will have low doping. The transistor formed by the right side of D2 and S will have higher doping. If the doping is linear, the doping of the left channel will differ from that in the center of the structure by an amount equal and opposite to the amount of change in the right channel of that transistor. Their average value remains constant and equal to that of the transistor between S and D1. Thus, the two transistors (D1 to S, and D2 to S) will be matched. Right: Photolithographically invariant layout compensates for non-vertical dopant implantation, not uncommon in drain/source implants. If dopants are implanted from a source to the right of the shown transistor, each gate can cast a shadow wherein fewer ions are implanted to the left of the gate. This results in capacitance mismatch between C_{GS} and C_{GD}. Laying out each transistor as two oppositely oriented transistors, the effect of the implant angle is matched.

important topic continues to advance.

In our design flow, we simulated the effects of mismatch on our circuits prior to fabrication. We used Pelgrom coefficients as provided by the foundry applied to the standard BSIM3V3.1 simulation model.

3.1.2 Mismatch-Invariant Layout

Mismatch-invariant layout [5] is an obvious technique to consider. Some mismatch is due to doping concentrations varying gradually across a chip. This can be counteracted by using common centroid layout, illustrated and explained in the left side of fig. 3.1. Another source of mismatch occurs if ions are implanted at an angle slightly off perpendicular to the chip. The transistor gate acts as a shield and casts a shadow on the side opposite the ion source, causing differing impurity concentrations in the source and drain diffusions. Photolithographically invariant layout addresses this issue, as shown in the right side of fig. 3.1. Small scale local variations in doping or imprecise sizing due to imperfect lithography can be compensated by simply making the transistors larger, thus averaging over a greater area and decreasing the percent error. Lastly, the surround of a device impacts its operating characteristics. For good matching within arrays, dummy devices can be used to surround functional devices such that all functional devices are similarily influenced by abutting structures.

3.1.3 Mismatch in Arrays

Unfortunately, layout techniques are inadequate for reducing mismatch in the circuits in the pixel array discussed in this thesis. Common-centroid layout can reduce mismatch among a small number of nearby transistors. However, it is not applicable to reducing mismatch in circuits at distant ends of a chip, such as the many pixels in an array. Photolithographically invariant layout could be applied but the magnitude of the improvement would likely be insufficient. Simply resizing critical transistors to be larger is impractical for circuits embedded within pixels. Performing any computations within a pixel inevitably reduces the fill factor (percentage of a chip devoted to light sensing). Enlarging the computational circuits to improve matching would further reduce the sensor resolution given a set area of silicon, ultimately limiting the number of cells in the array to an impractically small number. Mismatch between devices has previously been a critical barrier to producing truly useful arrays of local processing elements [62], especially in design using subthreshold transistors [20]. Thus, we must look to some other technique to address this problem

3.1.4 Reducing Mismatch with Floating Gates

Floating gate technology is used here as a compact means of reducing mismatch in circuits after fabrication [25]. Floating gate devices are a form of non-volatile memory, which means they are able to store their values for extended periods of time (on the order of years) as well as when power to the circuit is turned off. The devices are programmed once to permanently remove mismatch between the subcircuits into which they are embedded. Unlike EEPROMs which are programmed to either an *on* or an *off* state, the floating gate transistors used herein are programmed to analog values. Floating gate transistors are widely used in digital memory storage, namely in EPROM and EEPROM, whose applications include CompactFlash cards and the BIOS of a computer. However, floating gate use in analog circuits remains within the toolbox of a much smaller group of designers. This thesis describes circuits which are an interesting addition to that toolbox. I have embedded them in the logarithmic photoreceptor, in a novel difference circuit, and in the circuit which determines saliency of image features.

The cross section of a floating gate transistor is shown in fig. 3.2. Unlike a standard MOSFET, the gate is not electrically connected to any other circuit nodes. Since the gate is fully surrounded by silicon dioxide, data retention times are vastly longer than those of capacitors. For processes with gate oxide thicknesses exceeding about 70Å, the retention time is measured in years or decades. Thus, we are able to program the floating gates to remove offset initially and operate the chip without further reprogramming. The control gate, if fabricated, couples capacitively to the floating gate. A transistor with a control gate functions much like a standard transistor with two exceptions. First, the charge on the floating gate adds an offset to the threshold voltage. Second, because the

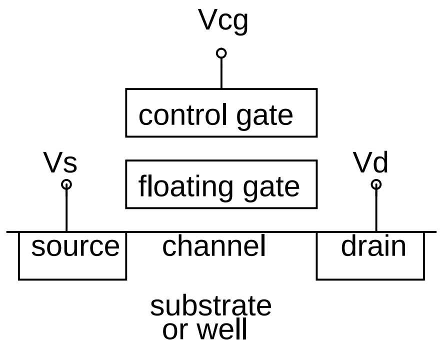


Figure 3.2: Cross section of a floating gate transistor (not to scale).

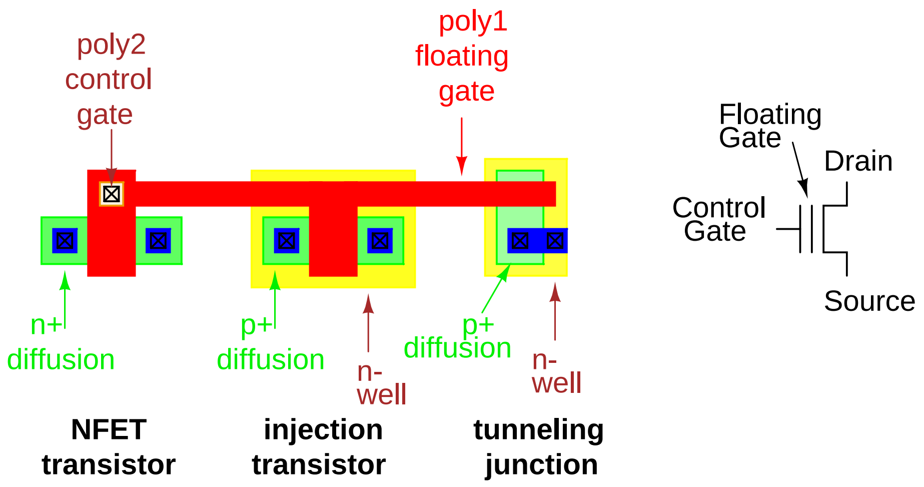


Figure 3.3: NMOS floating gate transistor, shown with structures for injection and tunneling. The circuit symbol for the NFET floating gate transistor is shown on the right.

control gate couples to the channel via the floating gate, the coupling between the control gate and channel is weaker than in a standard transistor, and thus the gain is lower. To calibrate the device to its desired operating point, charge is added to or removed from the gate by hot carrier injection or tunneling, respectively. In my circuits, tunneling is used globally to erase all floating gates. Injection programs devices individually. The resulting analog value stored on the floating gate can be thought of as either setting the current through the device to a chosen value, or adding an offset to the transistor's threshold voltage. The devices are programmed to values such that the circuit's overall operating characteristics match some desired goal.

Example layouts of floating gate devices are shown in figs. 3.3 and 3.4, including supporting circuitry for erasing and programming. Note that the floating gate is not connected to any external nodes. The left-most device in the two figures is the floating gate transistor. The control gate is optional, although useful in experimental circuits for debugging purposes. The *injection transistor*

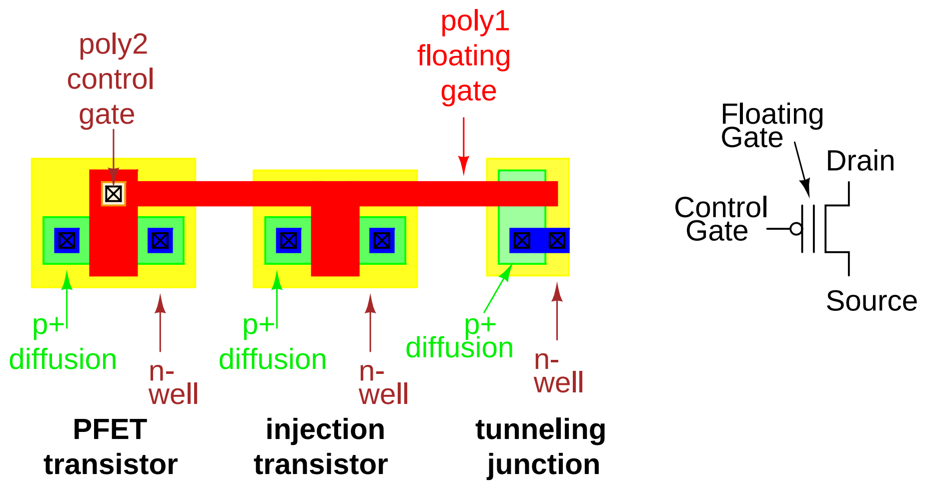


Figure 3.4: PMOS floating gate transistor, shown with structures for injection and tunneling. The circuit symbol for the PFET floating gate transistor is shown on the right.

uses hot carrier injection to deposit electrons onto the floating gate. In NFET transistors, channel electrons are injected directly onto the gate. In PFET transistors, channel holes generate electron-hole pairs, and these secondary electrons are then injected onto the gate. The *tunneling junction* uses Fowler-Nordheim tunneling to remove electrons from the gate. In modern processes with thinner oxides, direct tunneling may also occur. A more detailed discussion of injection and tunneling can be found in [24].

When floating gate devices are expected to retain charge over long time periods, leakage through the oxide is a concern. Charge can escape via direct tunneling. This process is exponential in the thickness of the gate oxide. At gate thicknesses at or above 70Å, retention times are in years or decades, depending on the desired precision. Below 70Å, leakage due to direct tunneling across the gate oxide increases exponentially, limiting the scaling of oxide thicknesses and thus transistors to be used in such memories. This trend is evident in the data of fig. 3.5. If we wish to fabricate analog floating gate memories, we will run into the leakage problem. However, many modern processes provide two types of transistors to circuit designers. In addition to the thin-oxide transistors scaled as in fig. 3.5, the processes provide thick-oxide transistors intended for interfacing with higher voltage devices off chip. These thick-oxide devices often have oxides thick enough for good retention times and can be used within circuits wherever non-volatile analog (or digital) storage is useful.

3.2 Pixel Overview

The mathematical derivation of the Orthogonal Gradient Detector (OGD and comparison of the OGD to the Kanade detector are presented in sec. 2.4, concluding with an algorithmic simplification.

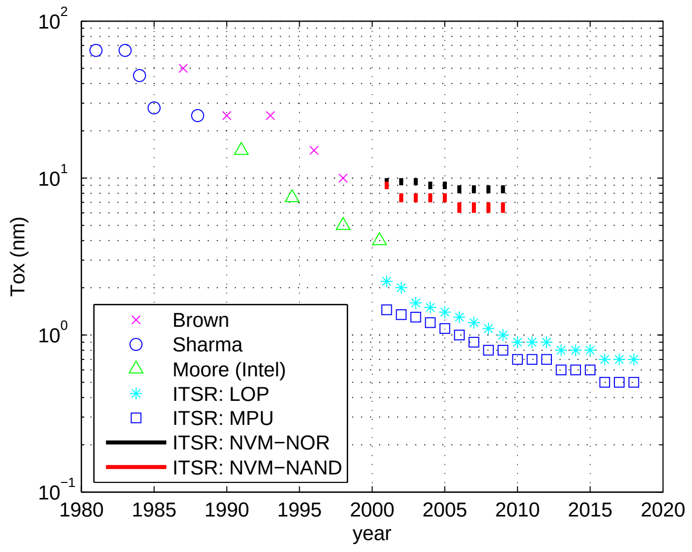


Figure 3.5: Gate oxide thickness as a function of time. The date is extracted from two books (by Brown [64] and by Sharma), a talk by Gordon Moore (Intel), and the International Technology Roadmap for Semiconductors [1]. LOP = "Low Operating Power." MPU = "Microprocessor Unit," referring to high-performance digital logic. NVM = "Non-Volatile Memory." While the exact values are open to discussion based on whether one prefers to note cutting edge technology in development or well-established processes, we can note some trends. First, as lithographic transistor dimensions are scaled down every year, oxide thicknesses also decrease monotonically. Second, the oxide thicknesses for non-volatile memories plateau at about 70Å.

This section discusses the VLSI implementation of the OGD.

Each pixel detects light and calculates the horizontal and vertical gradients at that point based on its neighbors' photoreceptor outputs. It stores these gradient values for two time steps, chosen by an external clock. During readout, each pixel's gradient values are thresholded to yield a determination of whether that pixel is centered on a salient feature or not. This saliency calculation is performed by a single on-chip circuit that is serially shared by all pixels. Digital processing external to the chip can estimate local motion at each salient pixel. The salient pixel's gradient values at the current time step can be compared with values stored by neighbors at the previous time step to determine where in the image the salient feature was located previously.

The functionality encapsulated by each pixel is shown schematically in fig. 3.6. Each pixel contains a photoreceptor. Its output (L) is proportional to the logarithm of the light intensity and provided as a voltage input to each of its neighbors. The pixel performs two difference operations, one to calculate the horizontal gradient ($L_{UP} - L_{DOWN}$) and a second to calculate the vertical gradient ($L_{LEFT} - L_{RIGHT}$). Four sample-and-hold (S/H) circuits store the gradient values for the current and previous time steps, for each of the two difference calculations.

In practice, the pixel is implemented slightly differently, as shown in fig. 3.7. Since each difference circuit (sec. 3.4) contains two floating gates, including two copies of the difference circuit

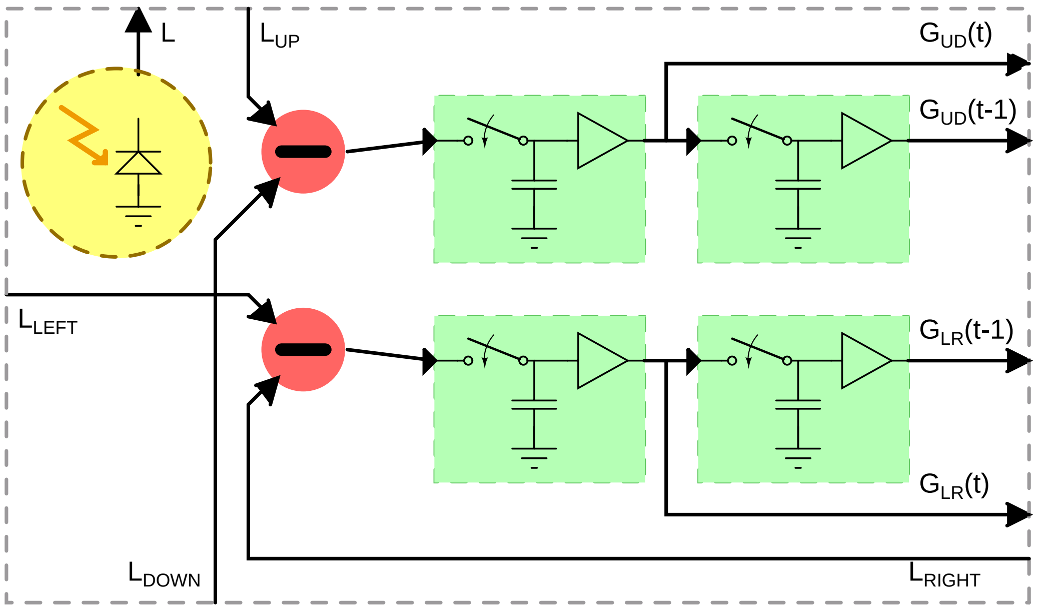


Figure 3.6: Conceptual block diagram of a single pixel. L denotes the photoreceptor output in response to the sensed light intensity. The analogous responses of the pixel's neighbors are denoted as L_{UP}, L_{DOWN}, L_{LEFT}, and L_{RIGHT}. The outputs of the difference circuits are stored in sample-and-hold circuits. Since they represent the vertical (Up-Down) and horizontal (Left-Right) gradients, they are denoted as G_{UD} and G_{LR}, respectively. The pixel stores gradient values for the current time step as $G_{UD}(t)$ and $G_{LR}(t)$; and for the previous time step as $G_{UD}(t-1)$ and $G_{LR}(t-1)$.

would increase the layout area much more than the alternative implementation. In the alternative implementation, the inputs from the neighboring pixels are switched such that the difference circuit calculates first the horizontal gradient, and second the vertical gradient. The area of the switches is less than that of a second difference circuit.

In-pixel processing is appropriate for local operations that map efficiently into circuits. This is an example wherein a simple and local calculation is efficiently implemented on the same chip as the photo-sensing array, parallelizing a huge computational task and vastly reducing the amount of data to transmit off chip.

3.3 Photoreceptor Circuit

The light sensing element that serves as the input to each pixel is implemented as a logarithmic continuous-time photoreceptor based on a photodiode. Before delving into the details of this photoreceptor, I present a brief overview of silicon photodetectors to give context to the one I built. Sec. 3.3.1 mentions several types of photo-sensitive devices. Sec. 3.3.2-3.3.3 present an overview of some common photoreceptor topologies based on photodiode devices. Sec. 3.3.4 discusses a sampling of previously fabricated photoreceptors with mismatch compensation. Finally, sec. 3.3.5 presents

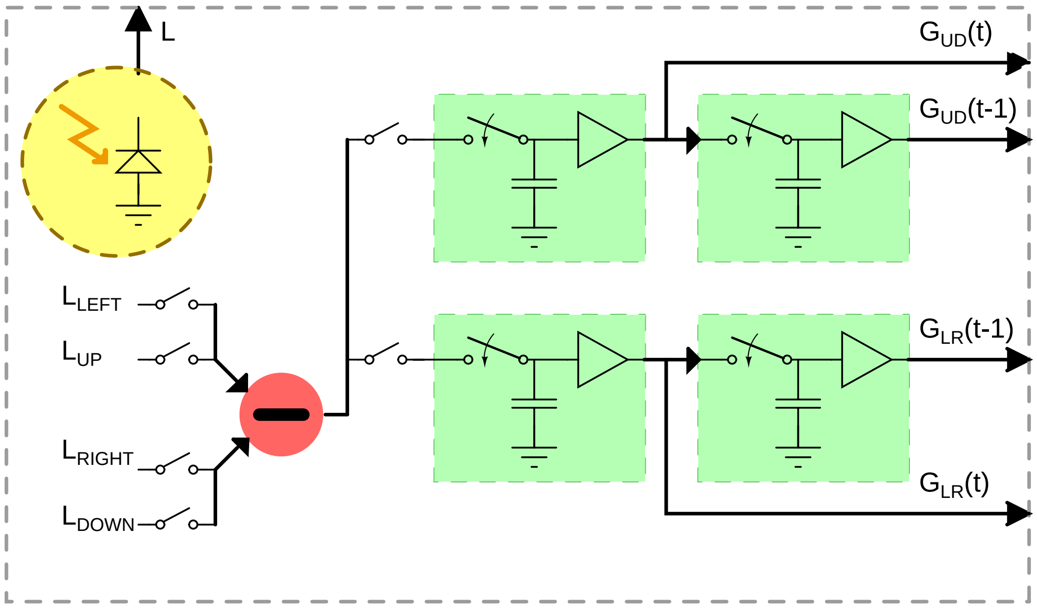


Figure 3.7: Block diagram of a single pixel, as implemented. To reduce layout area, only one difference circuit is used. Switches are used to select between inputs (either horizontal or vertical) and which sample-and-hold should store the difference.

my photoreceptor, along with measured data.

3.3.1 Light Detection Devices

Silicon photodetectors are implemented by means of one of several devices, whose photoresponse is based on the same underlying principles of silicon physics. The most common devices used for photodetection include photodiodes, charge-coupled devices (CCDs), and phototransistors, both MOS and bipolar. Sze [74] provides a good introductory review of these devices. A photodiode can be implemented (as in this thesis) as a lightly-doped n-well in a lightly-doped p-substrate. The p-substrate is grounded while the n-well is held at a positive potential, reverse-biasing the diode. When a photon strikes the silicon lattice in or near the diode's depletion region, an electron-hole pair is generated. The applied voltage sweeps the electron and hole in opposite directions, generating a measurable current, the *photocurrent*. In this thesis and in other CMOS sensors, a small circuit typically converts the current into a voltage or somehow amplifies the photocurrent before off-chip readout. A CCD array is basically an array of closely spaced MOS diodes. Each device is reset to some voltage and then photocurrent causes accumulation of charge in the CCD. For readout, the last CCD in each row is read. Then the packets of charge are shifted to their neighbors, and the next row is read. Specialized processes for fabricating CCDs allow this shifting of charge packets to be done with minimal loss. Because the fabrication process is specialized, circuits are not implemented on the same chip. For smart sensors that wish to incorporate some preprocessing, including windowing, calibration, feature detection, or random-access readout, the CCD process is not an option. CMOS processes allow fabrication of high quality circuits with embedded photodiodes or phototransistors.

3.3.2 The CMOS Integrating Pixel Sensor

Among CMOS image sensors, the most common technique for converting light into an electrical output is to integrate the photocurrent onto a capacitor for some set time period. Such sensors include those encompassed by the catch-phrase "Active Pixel Sensor" or "APS." As shown in fig. 3.8, a switch, implemented as either a PFET [22, 83] or NFET transistor, resets the output voltage node V_{out} to a positive reset voltage, commonly the power supply rail V_{dd}. Light induces a current through a photodiode, commonly formed by the junction of an n-well with the p-substrate. The photodiode has a significant capacitance. Thus, the photocurrent I drains the voltage $Vout$ from the capacitor C according to $I = C \cdot \frac{dV_{out}}{dt}$. The more light, the more photocurrent, and the more quickly $Vout$ drops. After some set integration time, the voltage on the capacitor is read out. The voltage drop $V_{dd} - V_{out}$ is proportional to the number of photons captured by the device. The integration time can be shortened at very bright light intensities and lengthened under dim conditions to extend the dynamic range of the sensor.

The choice of NFET versus PFET switch is a trade-off between layout size and voltage range. An NFET can be laid out directly in the p-substrate. PFETs require an n-well separate from the photodiode and well spacing rules force the PFET-switched pixel to be larger than its NFET counterpart. On the other hand, the maximum voltage to which the NFET version can be reset is lower than with a PFET. For an NFET, `Vout` is reset to `Vdd - Vth` where `Vth` is the transistor threshold voltage. With a PFET, that node can be fully reset to `Vdd`. As power supply voltages drop in modern processes, maintaining a reasonable output range may require choosing a PFET switch.

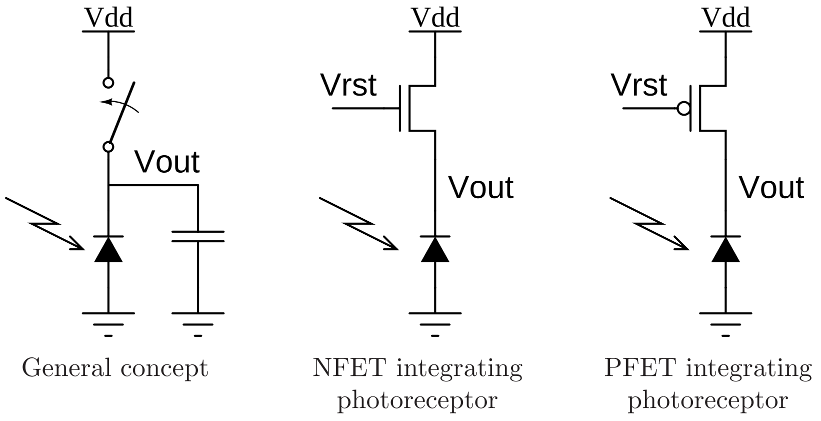

Vdd
Vout
General concept
Vdd
Vrst
Vout
NFET integrating photoreceptor
Vdd
Vrst
Vout
PFET integrating photoreceptor

Figure 3.8: The basic topology of an integrating photodiode sensor. The capacitance of the NFET and PFET instantiations is provided by the inherent capacitance of the photodiode.

3.3.3 The CMOS Continuous-Time Logarithmic Photoreceptor

Logarithmic photoreceptors send the photocurrent generated by by the photodiode through a transistor operating below threshold. In subthreshold, the gate-source voltage of the transistor is exponentially related to the current so the photoreceptor performs a logarithmic compression on the signal. Such compression is analogous to human vision and extends the dynamic range of the photoreceptor far beyond that of a CCD camera [51].

Possibly the simplest logarithmic photoreceptors based on a photodiode consist of one photodiode and one transistor, either an NFET or a PFET [7]. The respective topologies are shown in fig. 3.9. Since the transistors in these photoreceptors operate below threshold, the voltage V_{out} is

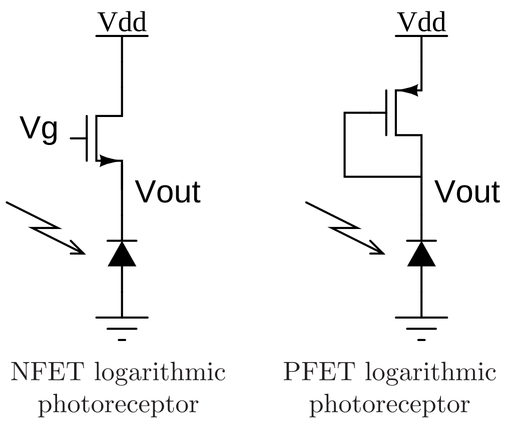

Vdd
Vg
Vout
NFET logarithmic photoreceptor
Vdd
Vout
PFET logarithmic photoreceptor

Figure 3.9: Simple logarithmic sensors based on photodiodes.

logarithmically related to the photocurrent I_{ph}:

$$\text{NFET:} \quad I_{ph} \propto e^{V_g - V_{out}}$$
$$\text{PFET:} \quad I_{ph} \propto e^{V_{dd} - V_{out}}$$

In the case of the NFET, the gate voltage V_g may be tied to V_{dd} for simplicity. Alternately, V_g can be set by a bias to allow the output voltage to be shifted. If V_g is shifted down by some amount, V_{out} will move down by the same amount.

Feedback may be used to increase the speed of the sensor [50, 21]. In the non-feedback topologies shown in fig. 3.9 after a sudden change in the light intensity the photocurrent must fully charge the capacitance of the photodiode to a new value before V_{out} reflects the change in illumination. Since the photocurrent may be very small, at dim light levels this charging time limits the temporal bandwidth of the photoreceptor. On the other hand, in the topologies shown in fig. 3.10, the current driving the photodiode node is sourced by an amplifier and can be chosen to give a reasonable bandwidth even at low illumination.

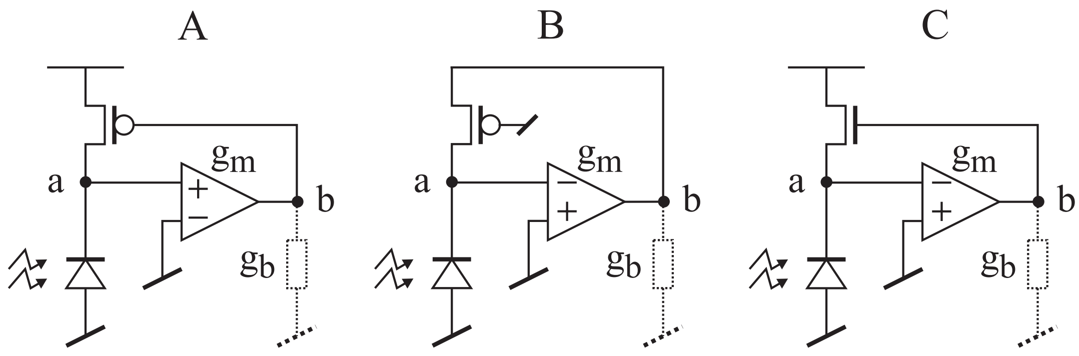


Figure 3.10: Logarithmic photoreceptors with feedback. The topology shown in A was used by Ni et al. [58]. Landolt et al. [50] chose topology B after analysis of all three topologies for bandwidth and damping. Delbrück et al. [21] implemented a topology similar to C. The transconductance of the amplifier is denoted by g_m. Most simply, the amplifier can be implemented as a single transistor with node a attached to the gate and the source and drain connected to node b and ground. The output resistance is denoted by g_b. When the circuit is not driving a resistive load, g_b arises from the amplifier.

3.3.4 Mismatch and Calibration

The problem of mismatch is pronounced in analog focal plane processing arrays due to their use of logarithmic photodetectors. Integrating pixel sensors, which reset a photodiode and sample it some time later as explained in sec. 3.3.2, are affected primarily by mismatch between the photodiodes. In modern high-quality processes, this mismatch is low enough to be acceptable for many applications. Measurements in 1.2μm and 2.0μm processes [46] indicate less than 2% mismatch between photodiodes. Unfortunately, sampled photoreceptors are not well suited to continuous-time analog computation. Mismatch in logarithmic photoreceptors, introduced in sec. 3.3.3, results primarily from voltage threshold mismatch in the transistor that performs the logarithmic transformation. Because mismatch between transistors is larger than that between photodiodes, logarithmic photoreceptors suffer from much greater mismatch than integrating photoreceptors.

Fig. 3.11 illustrates the effects of mismatch on a linear array of 16 logarithmic photoreceptors fabricated in a 0.35μm process. I uniformly illuminated the array and swept the illumination over a wide range. Note that the constant slope of the photoreceptors. Most of the mismatch results in offset error, which is what we should expect of subthreshold transistors whose thresholds are poorly matched. Note also that this mismatch-induced shift in output voltage is of a magnitude comparable to that resulting from typical signals. For example, the change in output voltage corresponding to the difference between viewing an LCD monitor at its brightest setting and its darkest setting is 135mV. For a single light intensity, the 16 photoreceptors output voltages ranging over 60mV with a standard deviation of 15mV, 44% and 11% of the LCD monitor signal range, respectively. Some sort of mismatch correction, either within the photoreceptor array or in later processing, is critical to fabrication of functional logarithmic imagers [51, 61].

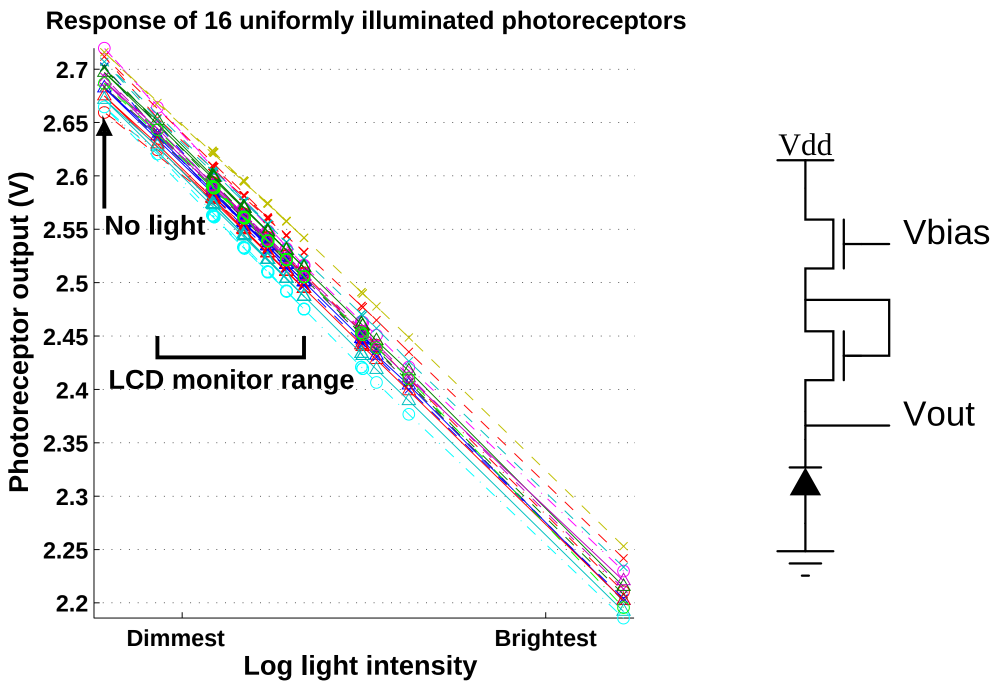


Figure 3.11: I uniformly illuminated a linear array of 16 logarithmic photoreceptors. As I swept the magnitude of the illumination, the voltage output of the photoreceptors changed with the log of the light intensity, as expected.

Samples of previous work on mismatch reduction in photoreceptors are summarized in Table 3.1. Kavadias et al. [47] perform compensation for fixed-pattern noise (FPN) in a manner akin to correlated double sampling in integrating CMOS sensors. That is, they subtract the output of the pixel in response to the photocurrent from the output in response to a calibration current. In a different approach, Loose et al. [51] built a self-calibrating camera with logarithmic photoreceptors. The photoreceptor topology is similar to the NFET logarithmic photoreceptor shown in fig. 3.9. A capacitor in every pixel serves as an analog memory to store a calibration value for V_g to reduce fixed pattern noise (FPN). Because capacitors leak charge, the stored voltages must be refreshed periodically and calibration is performed every frame. Unlike capacitors, floating gate memories (Electrically Erasable Programmable Read-Only Memories, EEPROMs) allow non-volatile storage. The floating-gate calibration proposed by Aslam-Siddiqi et al. [6] calls for one-time calibration with no further updates necessary during operation.

3.3.5 The Fabricated Logarithmic Floating-Gate Photoreceptor

The sensor presented in this thesis is shown in fig. 3.12D. It uses a basic logarithmic topology with an NFET providing the logarithmic compression. Unlike its non-programmable counterpart (fig. 3.12A), it incorporates a floating gate to allow for one-time calibration to remove mismatch.

first author	year	process feature size (μm)	pixel size ($\mu m \times \mu m$)	photo element	calibration technique	remaining mismatch
Aslam-Siddiqi [6]	1998	1.5	31.2×763	PFET	floating-gate	2.5% of output range
Poliquen [66]	1999	0.5	40×50	PNP	capacitors for offset adjustment	var = 6% mean
		2.0	144×144			
Kavadias [47]	2000	0.5	7.5×10	diode	subtract response to ref. current	2.5% of output range
Loose [51]	2001	0.6	24×24	diode	in-pixel capacitor-based memory	not available
Cohen [16]	2001	1.2	34×34	PNP	floating-gate for gain adjustment	not available

Table 3.1: Overview of some previous implementations of logarithmic CMOS sensors with reduced fixed-pattern noise (FPN).

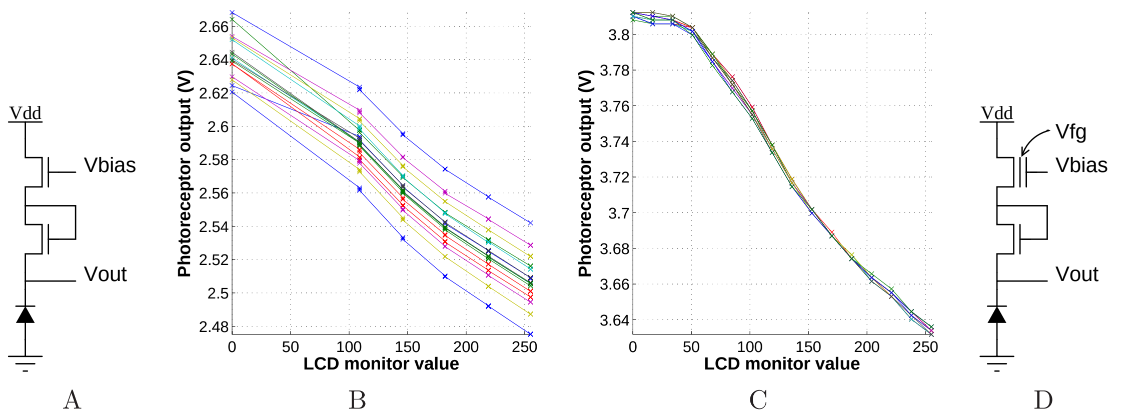


Figure 3.12: Logarithmic photoreceptor, with and without a floating gate for reducing mismatch, both fabricated in the same 0.35μm process. A: A basic logarithmic photoreceptor. B: The response of 16 non-programmable logarithmic photoreceptors to a uniformly illuminated LCD monitor. C: The response of 16 floating-gate logarithmic photoreceptors to the same stimulus. D: A floating-gate logarithmic photoreceptor.

The mismatch is vastly reduced, as shown in fig. 3.12B–C. In the non-adjustable version, the two worst-matched photoreceptor outputs differ by 60mV. The outputs of the sixteen photoreceptors vary with a standard deviation of 15mV. In the floating-gate version, the mismatch drops to a standard deviation of only 0.5mV at the light intensity at which I performed the calibration (monitor value of 177), and to a 1.5mV standard deviation when the standard deviations are averaged over all stimuli. Both values for the floating-gate photoreceptor are less than the 2.1mV resolution of the ADC (Analog-to-Digital Converter) in the instrument used to collect the data. Data from a larger array of 256 pixels (16×16) is shown in fig. 3.13. I excluded the edge pixels of the complete 18×18 array. After programming, the mean standard deviation among the 256 pixels is 5.2mV. Considering that the ADC converter used for feedback during calibration and for the measurements has a resolution of 2.1mV, this is quite reasonable. More precise programming is possible, limited here primarily by equipment and noise in the test setup.

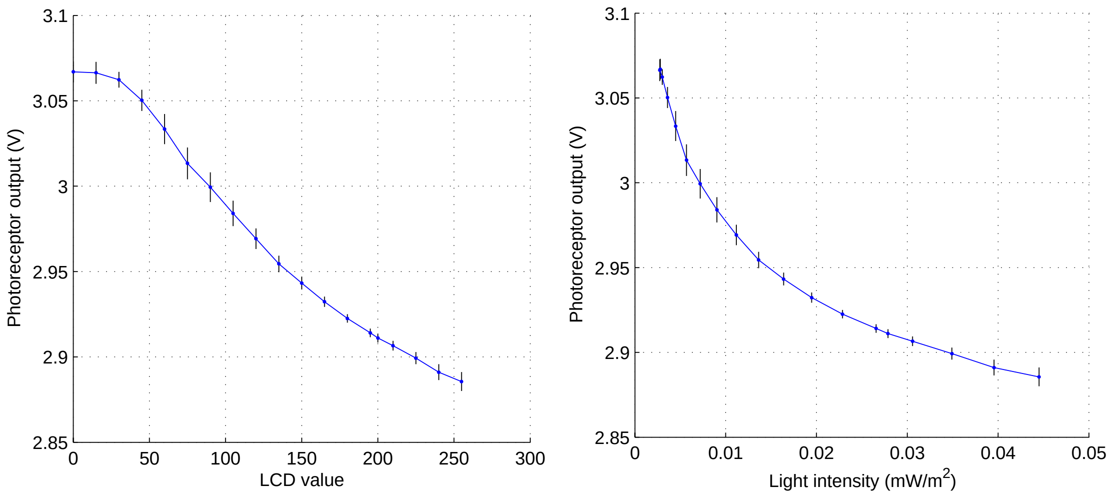


Figure 3.13: Floating-gate photoreceptor output voltage as a function of the illumination for a 16×16 array. Bars show the standard deviation of the pixel values at each point. Left: Data as a function of the LCD monitor value, in units used by the monitor. Right: The same data with the x-axis converted to light intensity, based on data from a light meter.

3.4 Difference Circuit

Having measured the light intensity at each pixel, we now need to extract the gradient of the light intesity along both the x and y axes. An analog difference circuit calculates this quantity. Unfortunately, analog circuits are impacted by transistor mismatch when small devices are employed. To remedy this problem, we incorporate two floating-gate transistors into each difference circuit. The floating gates are programmed to analog values, thus biasing the circuit in a manner that nulls process-dependent mismatch. An overview of the circuit is shown in fig. 3.14. V_1 and V_2 are inputs from neighboring pixels and correspond to light intensities. Floating gates fgγ and fgρ store precise quantities of charge which can be fine tuned individually for each difference circuit by injection and tunneling. The output of the difference circuit is a current. Due to the requirements of succeeding stages, we convert the current into a voltage. This voltage output is available for external measurement and all presented data are derived from measurements of the output of the current-to-voltage converter, V_{diff} in fig. 3.14.

Sec. 3.4.1 presents the topology of the difference circuit, the mismatch compensation mechanism, and equations characterizing both the basic functionality and the mismatch compensation. Sec. 3.4.2 explains the current-to-voltage converter, a simple circuit necessary for interfacing with subsequent circuitry. Finally, sec. 3.4.3 presents measured results from difference circuits on the fabricated chip

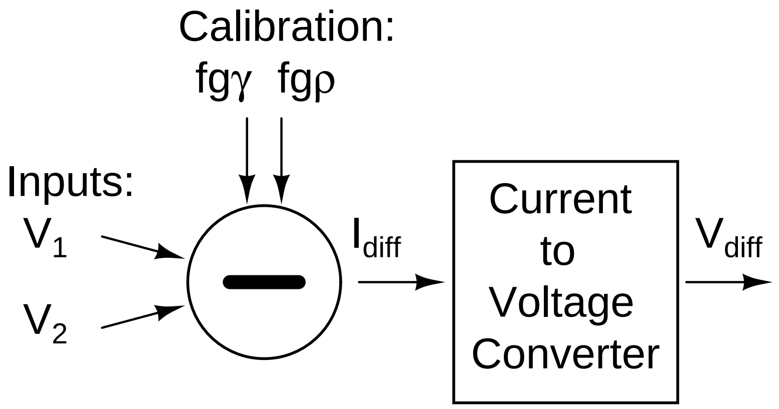


Figure 3.14: Overview of the difference circuit `diff`. Both the intermediate output I_{diff} and the final output V_{diff} are proportional to the difference of the inputs $(V_2 - V_1)$. In the context of the feature detection chip, V_1 and V_2 come from neighboring photodetectors such that V_{diff} is proportional to either the vertical or horizontal light intensity gradient.

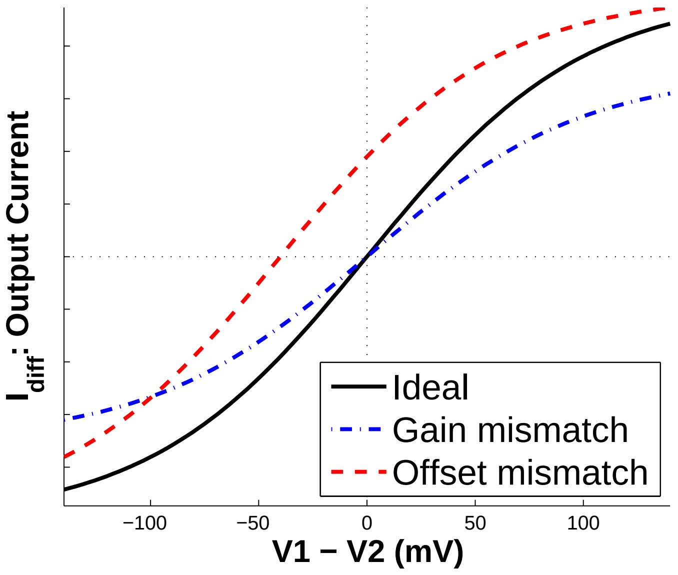


Figure 3.15: Cartoon illustrating the concepts of "gain" and "offset" as applied to the `diff` circuit. The output of the difference circuit should ideally be a line proportional to the inputs $(V_2 - V_1)$. The "gain" is the slope of this line, and the "offset" is how far the line is shifted from crossing the origin (0,0). Gain mismatch or compensation refer to changing the slope of the line, whereas offset mismatch or compensation refer to translating the curve.

and discusses deviations from ideality.

3.4.1 Difference Circuit Topology and Analytical Description

The difference circuit is based upon a differential pair whose two output currents I_1 and I_2 are subtracted by a current mirror, as shown in fig. 3.16. The operation of this circuit is described by:

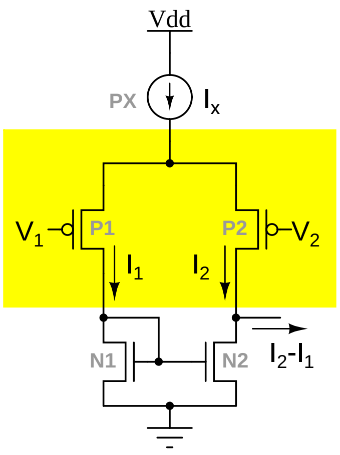


Figure 3.16: A differential pair constitutes the computational core of the difference circuit.

$$I_{diff} = I_2 - I_1 = I_x \; tanh\left(\frac{\kappa}{U_t}(V_1 - V_2)\right) \tag{3.1}$$

where U_t is the thermal voltage (25mV at room temperature) and κ expresses the effect that the transistor gate has on the surface potential in the channel, and is constant for a given process and transistor geometry. Since the *tanh* function is approximately linear in the region of operation, we can approximate eqn. 3.1 to:

$$I_{diff} = I_x \; \zeta(V_1 - V_2) \tag{3.2}$$

where ζ is a constant and I_x is set by the current source PX. As desired, the output current I_{diff} is linearly dependent on the difference of the inputs $(V_1 - V_2)$.

Since a line can be characterized by two parameters, and since this circuit is operated in its linear range, mismatch within the circuit also can be characterized by two parameters. I will refer to a shift in the slope of the line as a "gain" mismatch, denoted by γ. In physical terms, this corresponds to a change in the absolute magnitude of the output current. A translation of the line is denoted by ρ and referred to as an "offset" mismatch. "Offset" mismatch manifests itself as a non-zero output current when $V_1 = V_2$. Using this nomenclature, the effect of mismatch on the circuit can be expressed as:

$$I_{diff} = \gamma I_x \; \zeta(V_1 - V_2) + \rho \tag{3.3}$$

These concepts are expressed visually in fig. 3.15.

Two floating gate devices within the circuit allow compensation for mismatch along both degrees of freedom. To null gain mismatch, the current source for I_x is implemented by the floating-gate PFET PX shown in fig. 3.17. V_{bias} is an externally applied voltage shared among all difference circuits in all pixels. The precise amount of charge on the floating gate of PX (fgγ) sets the voltage $V_{fg\gamma}$, in turn setting the current I_γ. Since $I_{diff} \propto I_\gamma$, adjusting the charge on fgγ permits calibration of the output magnitude of I_{diff} or equivalently compensation for gain mismatch as expressed by the parameter γ of eqn. 3.3.

To compensate for offset mismatch, it is tempting to modify either P1 or P2 in fig. 3.16 to be a floating gate device. Consider making P1 a floating gate device. Changing the amount of charge on its floating gate is equivalent to introducing a shift in the threshold voltage, or adding a chosen offset V_{P1} to the input:

$$I_{diff} = \gamma I_x \; \zeta(V_1 + V_{P1} - V_2) + \rho$$

Being able to adjust V_{P1} would indeed permit nulling of ρ in eqn. 3.3 by setting $V_{P1} = -\rho/(\gamma I_x \zeta)$. Unfortunately, a floating gate device has a different transconductance than a regular transistor. Since the poly2 control gate must couple through the poly1 floating gate to the channel, the effect on the channel of a voltage change on the poly2 control gate is smaller than in the case of a regular transistor with a poly1 control gate. Expressing this change in transconductance, once calibrated the transfer function of the circuit would be:

$$I_{diff} = I_x \; \zeta(\kappa_1 V_1 - \kappa_2 V_2)$$

with $\kappa_1 < \kappa_2$. The inputs of the circuit (V_1 and V_2) would not be symmetric. To reintroduce symmetry, one could put a floating gate on each of P1 and P2 to equalize their transconductances. However, such circuits are difficult to program. Error in the output current can be corrected by either increasing the voltage on one floating gate or decreasing the voltage on the other floating gate. Finding a good common mode is difficult.

Rather than implementing P1 or P2 as floating-gate transistors, I modified the circuit topology as shown in fig. 3.17. In this solution, the gate voltages of all floating gate transistors remain constant once programmed. These transistors' transconductances thus do not affect the computation performed by the circuit. Transistor P1 of fig. 3.16 is duplicated and thus replaced by P1 and P4 of

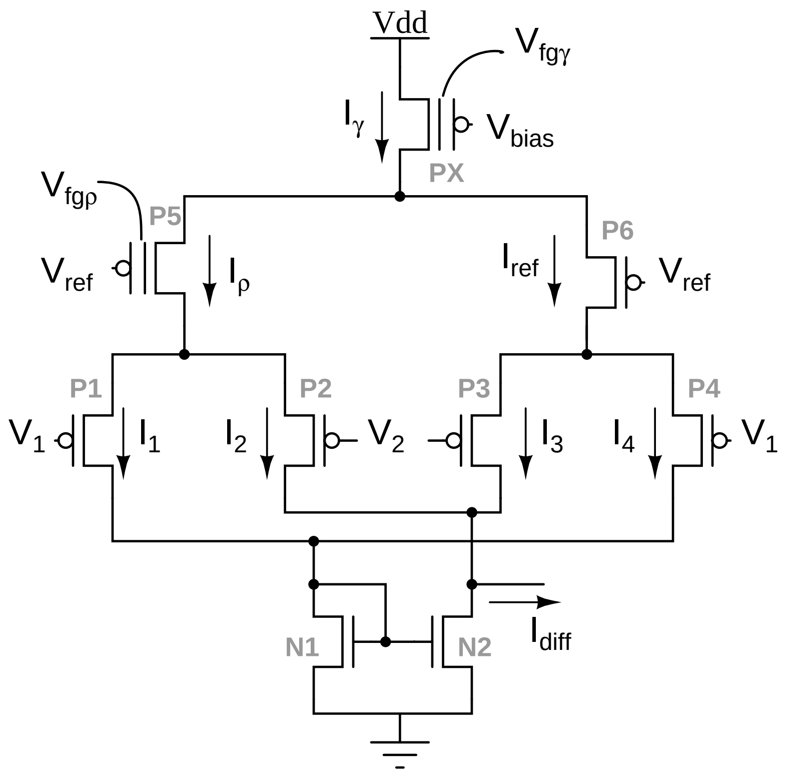


Figure 3.17: The topology of the fabricated difference circuit. Note that P1 and P3 are twice as wide as P2 and P4 to build an offset into each input differential pair, as discussed in the text.

fig. 3.17; similarily P2 is replaced by P2 and P3.

Ignoring mismatch for a moment, note that the transfer function of the circuit is unchanged compared to its simpler counterpart. The output current is $I_{diff} = (I_2 + I_3) - (I_4 + I_1)$. The operations of the differential pairs with V_1 and V_2 as inputs can be characterized by:

$$I_2 - I_1 = I_\rho \zeta(V_1 - V_2) \qquad \text{for P1 and P2}$$
$$I_3 - I_4 = I_{ref} \zeta(V_1 - V_2) \qquad \text{for P3 and P4}$$

We can combine these equations to describe the overall computation performed by the fabricated circuit of fig. 3.17:

$$\begin{aligned} I_{diff} &= I_\rho \zeta(V_1 - V_2) + I_{ref}\zeta(V_1 - V_2) \\ &= I_\gamma \zeta(V_1 - V_2) \end{aligned}$$

which is the same as eqn. 3.2 for the simpler circuit in fig. 3.16.

From here, we can discuss how mismatch will affect the circuit's transfer function and how

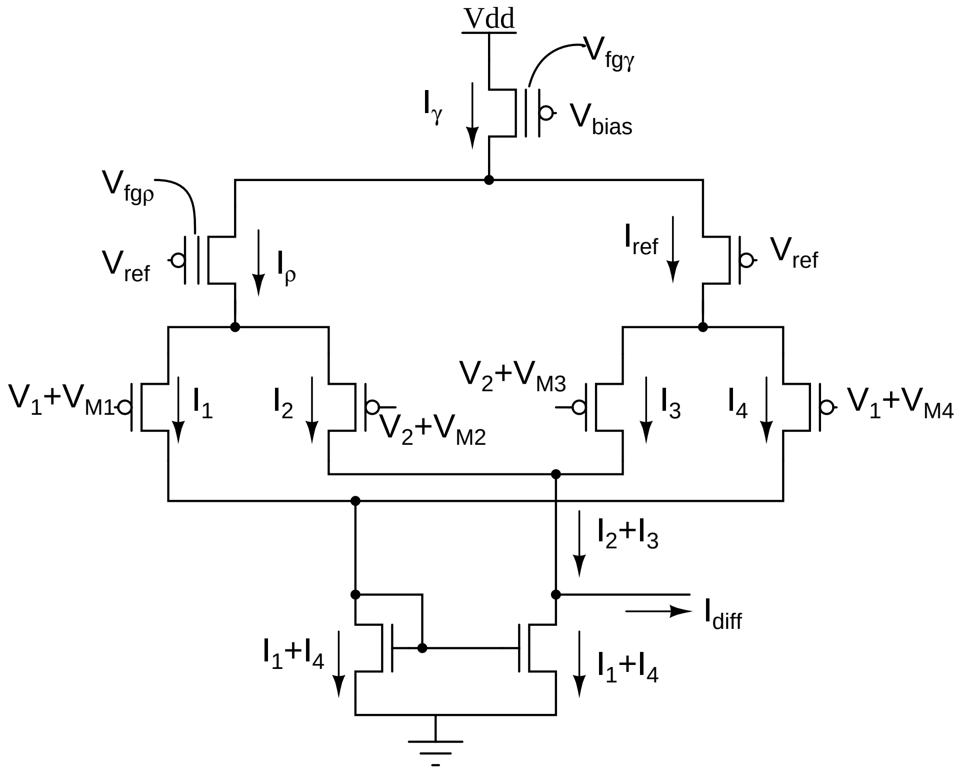


Figure 3.18: Mismatch in the difference circuit can be represented as offsets in the threshold voltages of the input PFETs. The threshold voltage shifts are denoted by V_{M1}, V_{M2}, V_{M3}, and V_{M4}.

programming floating gates fgρ and fgγ permits compensation for most of that mismatch. As previously explained, deviation from ideality in PX is directly addressed by adjustment of $V_{fg\gamma}$. Matching in P5 and P6 is only important inasmuch as it splits the current I_γ into I_ρ and I_{ref}, and directly addressed by programming fgρ. Of the remaining transistors, simulations suggest that mismatch in the input PFETs (P1-P4) is the dominant source of mismatch. We can represent the primary effect of this mismatch as a shift in the threshold voltage of transistors P1-P4. This is represented in fig. 3.18 by V_{M1}, V_{M2}, V_{M3} and V_{M4}. Mismatch in the current mirror (N1-N2) will affect the circuit but mismatch simulations indicated that the resulting error is minor. Without mismatch correction, mismatch in P1-P4 vastly overshadows that from N1-N2. Adjustment of $V_{fg\rho}$ changes the relative weighting of the input differential pairs (P1-P2 and P3-P4), correcting for offset error between these transistors.

We can now derive an equation characterizing the operation of the circuit, including mismatch in the input PFETs, and then derive the constraint that must be met to null the mismatch. The currents through the input PFETs are:

$$I_1 = \frac{1}{2}I_\rho + \frac{1}{2}I_\rho \cdot \zeta \cdot (V_2 + V_{M2} - V_1 - V_{M1})$$

$$\begin{aligned}
I_2 &= \frac{1}{2}I_\rho - \frac{1}{2}I_\rho \cdot \zeta \cdot (V_2 + V_{M2} - V_1 - V_{M1}) \\
I_3 &= \frac{1}{2}I_{ref} + \frac{1}{2}I_{ref} \cdot \zeta \cdot (V_1 + V_{M4} - V_2 - V_{M3}) \\
I_4 &= \frac{1}{2}I_{ref} - \frac{1}{2}I_{ref} \cdot \zeta \cdot (V_1 + V_{M4} - V_2 - V_{M3})
\end{aligned}$$

By Kirchhoff's Current Law, $I_{diff} = (I_2 + I_3) - (I_1 + I_4)$, which unifies the above equations to yield the output current:

$$I_{diff} = \underbrace{(I_\rho + I_{ref}) \cdot \zeta}_{gain} \cdot (V_1 - V_2) + \underbrace{I_\rho \cdot \zeta \cdot (V_{M1} - V_{M2}) + I_{ref} \cdot \zeta \cdot (V_{M4} - V_{M3})}_{offset\ due\ to\ mismatch} \tag{3.4}$$

We would like I_{diff} to be a linear function of $(V_1 - V_2)$ and include no mismatch terms. We would also like the gain term that defines the relationship between I_{diff} and $(V_1 - V_2)$ to be adjustable so we can match all difference circuits on the chip. In short, we would like to find parameters that return eqn. 3.4 to a form like: $I_{diff} = gain \times (V_1 - V_2)$. Regulating the gain is straightforward since $(I_\rho + I_{ref})$ is equal to I_γ, which in turn is directly programmable by fgγ. To remove offset due to mismatch, the last two terms in eqn. 3.4 must sum to zero:

$$0 = I_\rho \cdot \zeta \cdot (V_{M1} - V_{M2}) + I_{ref} \cdot \zeta \cdot (V_{M4} - V_{M3})$$

which is equivalent to the constraint:

$$\frac{I_\rho}{I_{ref}} = \frac{V_{M3} - V_{M4}}{V_{M1} - V_{M2}} \tag{3.5}$$

The current ratio I_ρ / I_{ref} is directly controlled by programming fgρ. Because both I_ρ and I_{ref} are unidirectional, the left-hand side of eqn. 3.5 will always be positive. To satisfy the constraint embodied by this equation, then, the fraction on the right also must be positive. However, since each mismatch term is equally likely to be positive or negative, the right-hand side of eqn. 3.5 will only be positive in about half the cases. To guarantee that this fraction is always positive, we build in offsets such that $V_{M4} > V_{M3}$ and $V_{M2} > V_{M1}$ (or, $V_{M4} < V_{M3}$ and $V_{M2} < V_{M1}$). On the chip, P1 and P3 are twice as wide as P2 and P4. Denoting this built-in offset as V_{bi}, the transfer function of eqn. 3.4 becomes:

$$\begin{aligned}
I_{diff} &= I_\gamma \zeta (V_1 - V_2) \quad + I_\rho \zeta (V_{M1} - V_{M2} - V_{bi}) \; + I_{ref} \zeta (V_{M4} - V_{M3} + V_{bi}) \\
&= I_\gamma \zeta (V_1 - V_2) \quad + (I_{ref} - I_\rho)\zeta V_{bi} + I_\rho \zeta (V_{M1} - V_{M2}) \; + I_{ref} \zeta (V_{M4} - V_{M3}) \qquad (3.6)
\end{aligned}$$

and the constraint of eqn. 3.5 becomes:

$$\frac{I_\rho}{I_{ref}} = \frac{V_{bi} + V_{M3} - V_{M4}}{V_{bi} + V_{M1} - V_{M2}}$$

We must choose V_{bi} such that even for the worst cases of mismatch, the mismatch terms of eqn. 3.6 can cancel to zero:

$$0 \;=\; (I_{ref} - I_\rho)\zeta V_{bi} \;+ I_\rho\zeta(V_{M1} - V_{M2}) \;+ I_{ref}\zeta(V_{M4} - V_{M3}) \tag{3.7}$$

When this constraint is met, offset mismatch will be fully nulled. If we denote the maximum expected value of V_{M1}, V_{M2}, V_{M3}, or V_{M4} as V_{Mmax}, for the cases of mismatch that require the most extreme adjustment the constraint of eqn. 3.7 becomes:

$$\begin{aligned} 0 &= (I_{ref} - I_\rho)\zeta V_{bi} \;+ 2I_\rho\zeta V_{Mmax} \;+ 2I_{ref}\zeta V_{Mmax} && \text{CASE 1} \\ 0 &= (I_{ref} - I_\rho)\zeta V_{bi} \;- 2I_\rho\zeta V_{Mmax} \;- 2I_{ref}\zeta V_{Mmax} && \text{CASE 2} \end{aligned}$$

or more succinctly:

$$\pm\frac{1}{2}\cdot\frac{I_\rho - I_{ref}}{I_\rho + I_{ref}} \;=\; \frac{V_{Mmax}}{V_{bi}}, \tag{3.8}$$

The two cases are symmetric and differ in whether $I_\rho > I_{ref}$ or $I_{ref} > I_\rho$, with the absolute value $|I_\rho - I_{ref}|$ being equal in both extreme cases. Recall that V_{Mmax} is the inherent process-dependent mismatch which as circuit designers we do not control, V_{bi} is the built-in offset chosen prior to fabrication, and I_ρ and I_{ref} are adjustable after fabrication. A proper choice of V_{bi} depends on V_{Mmax}, which we can only estimate, and the range over which we adjust I_ρ/I_{ref}. If $V_{fg\rho}$ is set significantly above (or below) V_{ref}, one of the branches of the circuit will turn off entirely, with $I_{ref} = I_\gamma$ and $I_\rho = 0$ (or $I_\rho = I_\gamma$ and $I_{ref} = 0$). This specifies the extremes of the range over which we can adjust fgρ and thus the ratio I_ρ/I_{ref}. Thus, the left-hand side of eqn. 3.8 can range between $\frac{1}{2}$ and $-\frac{1}{2}$. In order for the right-hand side to always remain within these bounds:

$$V_{bi} \geq 2V_{Mmax} \tag{3.9}$$

If the built-in offset V_{bi} is too small, not all mismatch in the offset will be removable. Since a prefabrication estimate of V_{Mmax} is indeed merely an estimate, it is prudent to make V_{bi} more than double the expected maximal mismatch V_{Mmax}. However, an overly large value of V_{bi} will require only very small adjustments in I_{ref}/I_ρ and thus very precise control of the charge on fgρ. Such precision in turn requires low noise on the supplies generating injection-related biases and careful design of the injection algorithm, both of which add to complexity and can lengthen the time required for calibration.

The calibration of the difference circuit can be divided into two phases, the first for fgρ and the second for fgγ. In the first phase, we adjust the ratio I_ρ/I_{ref} by injecting fgρ. A change on the gate of P5 (fig. 3.17) may cause a change in the voltage on the drain of PX, thus affecting the gain. Since the difference circuits are programmed one at a time, we temporarily adjust V_{bias}, if needed, during this phase of calibration to maintain a consistent gain. The second phase of calibration involves permanently adjusting the gain of the circuit. V_{bias} is returned to its base value and injection to fgγ brings the gain to its calibrated value. Simulation results in fig. 3.19 illustrate the calibration technique. Measured results are presented in sec. 3.4.3.

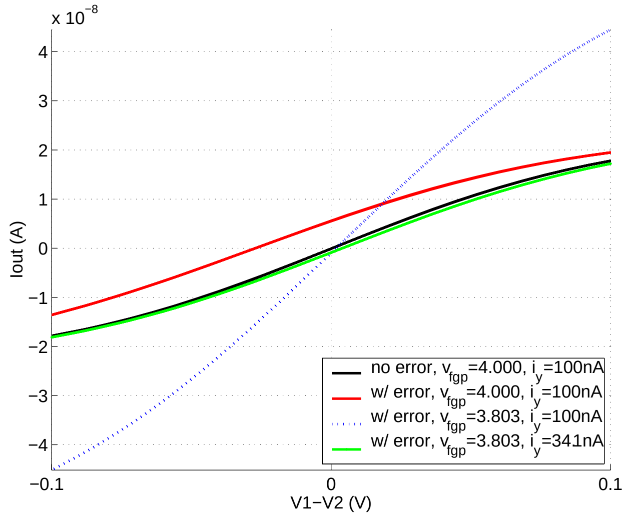


Figure 3.19: Simulation results for programming the difference circuit to remove mismatch. The black line shows initial simulation with no mismatch. Red line shows circuit output with error added to some of the inputs to model mismatch. Calibration phase 1: The dotted blue line corresponds to output after offset removal by adjusting $V_{fg\rho}$ and thus the ratio between I_ρ and I_{ref}. Calibration phase 2: The green line is current output after adjusting both $V_{fg\rho}$ and $V_{fg\gamma}$ to correct offset and gain, respectively. The gain is proportional to I_γ, which is controlled by $V_{fg\gamma}$.

3.4.2 Difference Circuit Current-to-Voltage Conversion

The output of the difference circuit is a current, yet a voltage is preferable as input to the succeeding stage. To detect motion, we need to compare image intensity gradients at one step in time with those at another step in time. To calculate features over the entire array for the same instant in time, we need to store the calculated gradients until they can be read off the chip. After considering a sample-and-hold (S/H) circuit topology with current input, we chose a voltage-input S/H to ease minimizing mismatch-induced errors within the S/H.

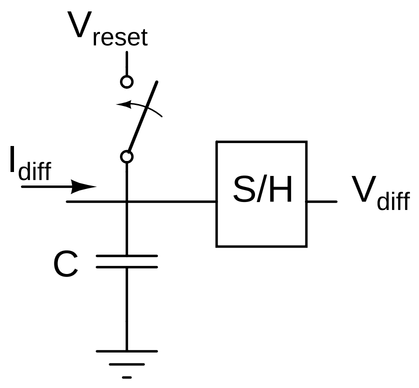


Figure 3.20: The current-to-voltage circuit integrates current onto a capacitor for a prespecified time interval. The resulting voltage is buffered and sampled by a sample-and-hold (S/H) which outputs a voltage that is linearly proportional to the input current.

The current-to-voltage converter is simply a capacitor, as shown in fig. 3.20. We reset the capacitor to some voltage V_{reset}. We then allow the difference circuit to output its current I_{diff} onto the capacitor for a set integration time t_{integ}. At the end of this time, we switch the S/H from sample mode to hold mode. The voltage is then available for use by the on-chip saliency circuit or for off-chip readout. The output voltage V_{diff} is linear in the input current. The capacitance C determines the scaling factor. The circuit is described by following equation:

$$V_{diff} \quad = \quad V_{reset} + I_{diff} \cdot \frac{t_{integ}}{C} \tag{3.10}$$

The range of the circuit is limited primarily by the limited range of our S/H circuit, approximately 2.25V to 4.25V. Since the output of the difference circuit should be symmetric around $I_{diff} = 0$, an intermediate value such as 3.25V is a good choice for V_{reset}.

All measured results from the difference circuit are buffered by the current-to-voltage converter and the sample-and-hold, as described here.

3.4.3 Difference Circuit Measured Results

Measured results from the difference circuit are presented in figs. 3.21 and 3.22. In each figure, the x-axis corresponds to the difference between the inputs V_1 and V_2. V_1 and V_2 are generated by photoreceptors, which in turn are activated by a visual stimulus generated by an LCD monitor. The y-axis is the current output of the difference circuit, I_{diff}. As explained in sec. 3.4.2, I_{diff} is not measured directly but rather calculated as dictated by eqn. 3.10 based on the measured voltage V_{diff}. V_{reset}, t_{integ}, and V_{diff} are measured quantities. C is estimated to be 282pF from the laid out area of the capacitor (a MOSCAP) and process parameters provided by the foundry.

Fig. 3.21 shows the shift in offset of the output of the difference circuit as $V_{fg\rho}$ changes due to

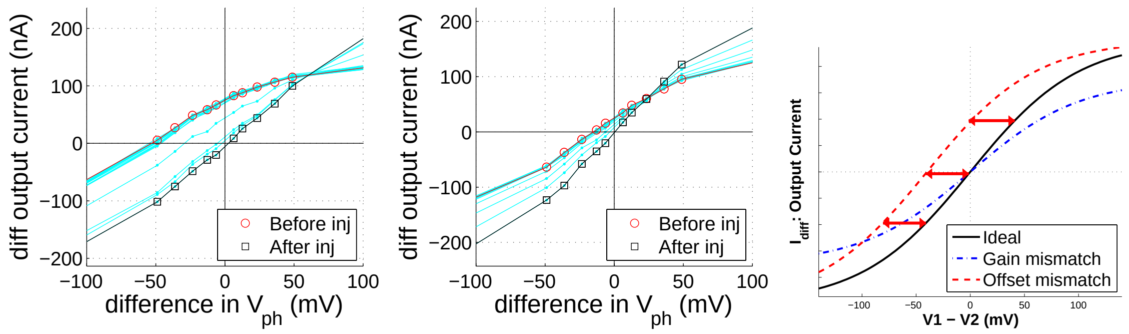


Figure 3.21: Injection to the floating gate fgρ changes the offset of the difference current output. Left and center: Measured data for two distinct difference circuits. Before programming, the data is offset due to mismatch (red circles). After programming, the transfer function is calibrated to cross the origin (black squares). Unmarked cyan lines indicate intermediate values during programming. Note that the gain is slightly affected by programming fgρ in both examples. Right: Cartoon of concept.

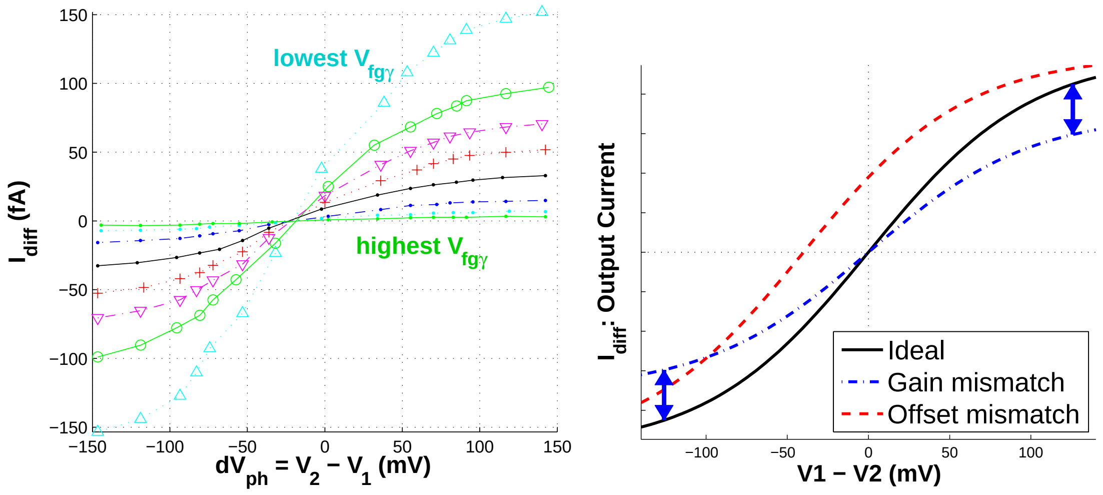


Figure 3.22: Sweeping the floating gate voltage $V_{fg\gamma}$ changes the bias current I_γ in the difference circuit, and thus the magnitude of the output current, allowing calibration of the *gain*. Left: Measured data. Right: Cartoon of concept.

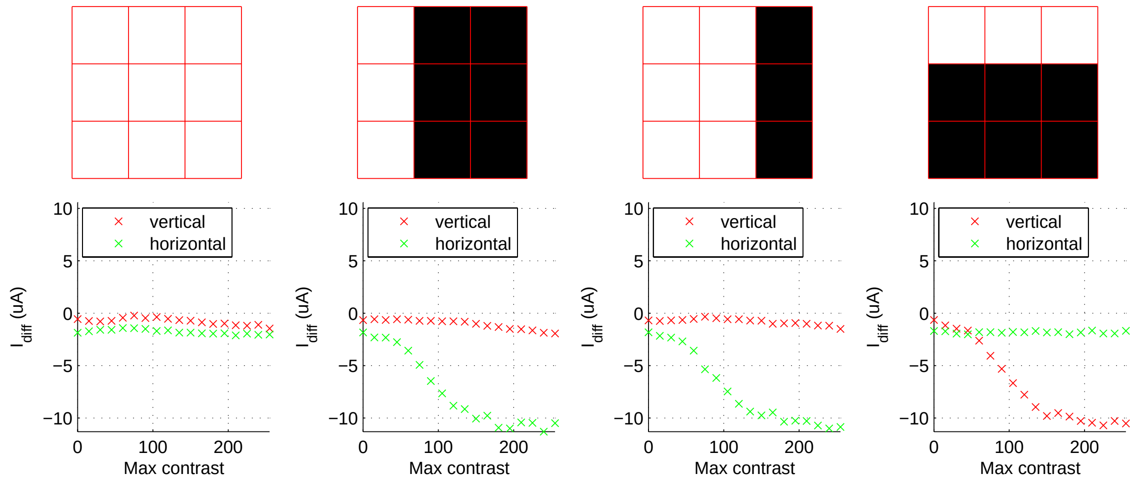


Figure 3.23: Output of difference circuit to blank screen and edge stimuli.

calibration by injection. Injection to the floating gate fgρ changes the offset of the difference current output. By precise injection, we can remove the offset such that the curve crosses the origin. In this case, a lower value of $V_{fg\rho}$ shifts the curve right. Whether the curve shifts left or right with injection depends on the mismatch. For example, we can see from eqn. 3.4 that if $V_{M1} > V_{M2}$ and $V_{M3} \approx V_{M4}$, increasing I_ρ will increase the output current shifting the curves left. If the mismatch is such that and $V_{M1} < V_{M2}$, an increase in I_ρ will have the opposite effect.

Fig. 3.22 shows the change in the gain of the difference circuit as the voltage on fgγ is changed. As $V_{fg\gamma}$ decreases, the bias current I_γ increases and the output current also increases. Numerical values for $V_{fg\gamma}$ are not provided because we cannot measure the floating-gate voltage. We can increase $V_{fg\gamma}$ by tunneling, decrease it by injection, and move it in either direction by sweeping voltages capacitively coupled to all the γ floating gates (V_{bias} and V_{tun}). For this figure, the light intensity was swept over the entire range of the monitor for the entire contrast range available from this stimulus.

Response of the difference circuit to visual stimuli is shown in figs. 3.23 and 3.24. Note that these results are for a poorly calibrated circuit, which is evident from the non-zero output in response to an input with zero gradient. For each stimulus, the contrast was varied from zero (white and black regions of the same intensity) to 255 (white and black regions maximally different, as permitted by LCD monitor stimulus). The responses are as expected. For edges, only one of the outputs changes with contrast, while the other gradient is calculated to be constant. For corners, both outputs change simultanously. For low contrasts, the LCD monitor range is compressed and the difference circuit does not detect a difference. For large contrasts, the difference circuit saturates because I used a constant-length integration time in the current-to-voltage conversion.

Array data are not available because not all structures can be programmed. In order to null

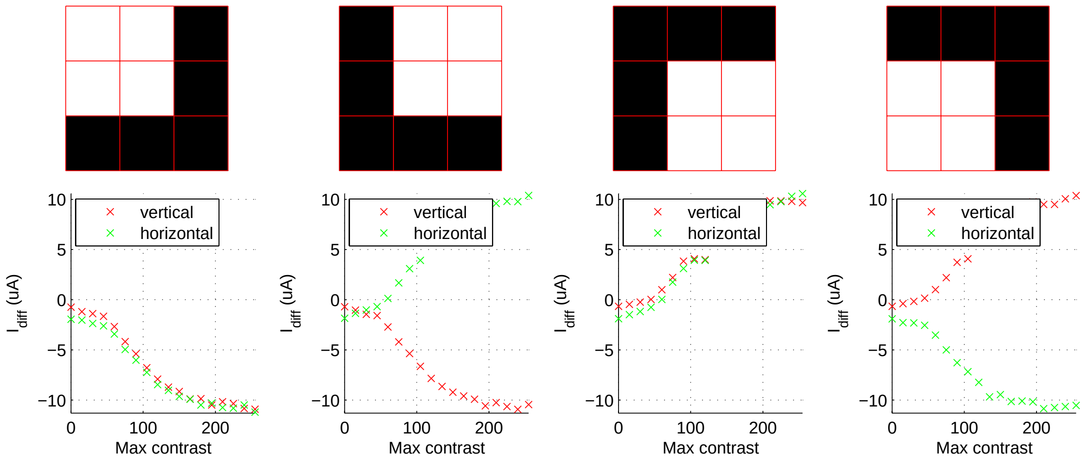


Figure 3.24: Output of difference circuit to corners.

mismatch after fabrication, the built-in offset in the input transistors (P1–P4) must meet the constraint of eqn. 3.9. Unfortunately, I underestimated the magnitude of the mismatch (V_{Mmax}) and did not build in a sufficiently large offset (V_{bi}). Approximately half of the difference circuits on the chip cannot be calibrated.

3.5 Sample-and-Hold Circuit

The voltage output of the difference circuit is stored locally prior to readout. We would like to capture information about gradients in the image at the same time for the entire image. If we stored the gradient and intensity information externally at time of readout, the value for the first pixel would be offset in time from that for the last pixel. More importantly, the information for a pixel on one row would be offset from that for its neighbors in other rows. Instead, a single complementary pair of clock signals initiates the transition from sample to hold for the entire array, assuring that gradient information for each pixel is captured at the same instant in time.

3.5.1 S/H Design

The chosen sample-and-hold (S/H) circuit topology can be described by the block diagram in fig. 3.25. The transistor level design is shown in fig. 3.26. Transistors P1-P5 and N6-N9 constitute the amplifier. The buffer is implemented as a source-follower with transistors P10 and P11. The MOS capacitor Ncap serves as the hold capacitor. The switches S12-S15 act to enable or disable the amplifier, thus transitioning between sample and hold modes, respectively.

In sample mode, transistor P1 provides the bias current for the amplifier. Switches S12-15 are as shown in fig. 3.26, setting their respective amplifier nodes to V_{pbias}, V_{pcas}, V_{ncas}, and letting the

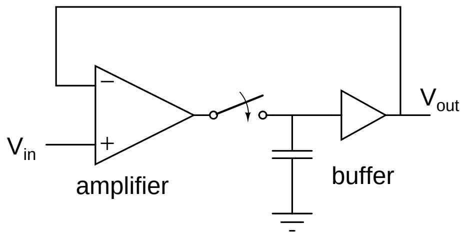


Figure 3.25: Block diagram of the sample-and-hold circuit.

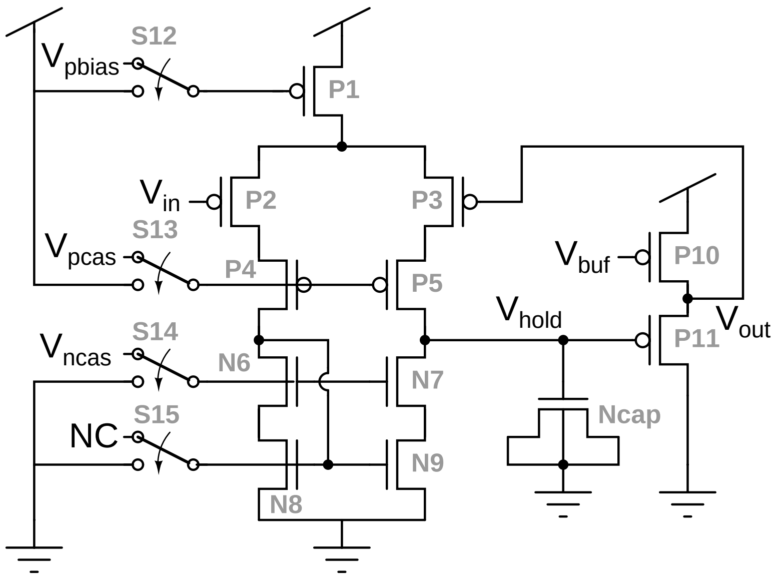


Figure 3.26: Transistor schematic of the sample-and-hold circuit.

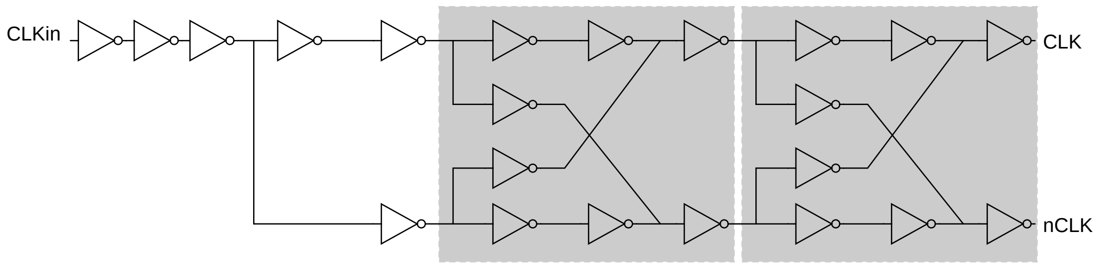


Figure 3.27: Generating a simultaneous pair of clock waveforms. Note that the circuit contains two identical stages in series. The delay between clk and $Nclk$ after the first stage is 14.6ps (simulated). The delay after the second stage is 0.8ps (simulated). In comparison, one inverter delay is about 134ps.

current mirror set itself. The input voltage V_{in} is buffered onto V_{out} by the combination of the amplifier and the buffer. The voltage on the hold capacitor V_{hold} is driven to some $V_{out} = V_{in}$ as needed to set the current through P11 equal to that sourced by P10. V_{buf} is generated by an on-chip bias generator and shared by all the S/H circuits on the chip. V_{buf} must generate a large enough bias current through P10 to keep the follower bandwidth high enough for stability in the feedback loop formed by the amplifier and the buffer. However, a large current through P11 forces V_{hold} lower. As V_{hold} drops, eventually Ncap goes into subthreshold and its capacitance decreases. This in turn increases the pedestal size, since the same amount of injected charge causes a larger voltage jump on a smaller capacitor. Ideally, then, V_{buf} would be just low enough to keep the S/H circuit stable, but no higher.

To switch to hold mode, switches S12-S15 are switched opposite to the position shown in fig. 3.26, setting their corresponding amplifier nodes to V_{dd} or V_{gnd} to turn off the amplifier. This turns off the bias current in the amplifier (P1), the currents through the cascode transistors (P4-P5, N6-N7), and the currents in the current mirror (N8-N9). Thus, voltages on most internal nodes of the amplifier should remain roughly where they were during operation, hopefully reducing leakage from V_{hold} through transistors P5 and N7 and decreasing the time needed for the amplifier to turn on when transitioning from hold to sample.

At the transition from sample to hold, all transistors in the amplifier need to turn off at the same time. The complementary clocks driving switches S12-S15 must be simultaneous. A single off-chip clock is provided and split into two complementary signals on-chip by the circuit shown in fig. 3.27. In simulation, the delay between the outputs clk and $Nclk$ is only 0.8ps, much less than necessary for the S/H to switch modes without a glitch.

A folded cascode topology was rejected for the smaller basic topology in fig. 3.26. While a folded cascode topology can improve range, the voltage of the preceeding difference circuit can be chosen to have a narrow enough range to allow for use of the non-folded topology. As a secondary benefit, the

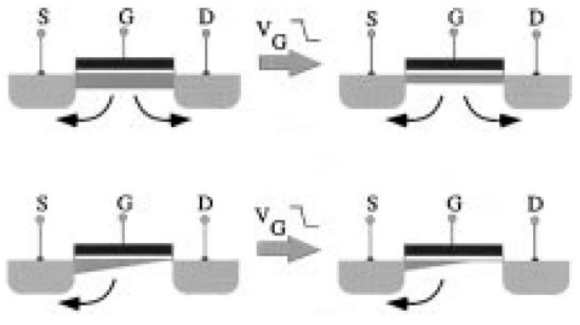


Figure 3.28: Channel charge in triode (top) and saturation (bottom).

non-folded topology has only four noise-inducing transistors, compared to six in a folded topology.

Consideration of stability is essential in the design of any feedback circuit. A hand calculation of the circuit in fig. 3.25, assuming two amplifiers and two capacitative nodes, suggests that the circuit should always be stable. However, simulation of the complete circuit in fig. 3.26 indicates the presence of a third significant pole. To maintain a phase margin of at least 60 degrees, we need to make sure that the buffer (P10-P11) is fast enough. This constraint requires a current of at least 450nA through these two transistors. The exact value of this current is controlled by V_{buf}. An on-chip bias generator produces V_{buf} through a series of current mirrors that rescale an off-chip reference current. P11 must be wide enough that even with this current, the gate (V_{hold}) will be at a reasonable voltage. If P11 is too narrow, the voltage V_{hold} necessary for P11 to sink its bias current may drop below the operating range of the amplifier or be insufficient to keep the channel of Ncap inverted. If this MOSCAP goes out of inversion, its capacitance will drop.

The cascode capacitors (P4, P5, N6, and N7) reduce charge injection onto the hold capacitor during the switch from sample to hold, as suggested by Dai and Harjani [18]. The cascode transistors are operated in saturation rather than in the triode region. In the triode region, channel charge is fairly evenly distributed throughout the channel, shown schematically in fig. 3.28. When the transistor turns off, some of this charge flows to the drain and some to the source. In saturation, the charge distribution is non-uniform with the majority of the charge near the source of the device. Thus, upon turning off, the source receives most of the injected charge. Only a much smaller fraction of the charge goes to the drain and thus the hold capacitor.

Most of the remaining pedestal should be a result of parasitic gate-to-drain capacitance in P5 and N7. This can be minimized by using narrow transistors for the cascodes. This effect is not symmetrical in P7 and N5, however. The cascode voltages V_{ncas} and V_{pcas} need to move a different amount to reach their respective rails (V_{dd} and ground) and switch off the cascode transistors. In simulation, values of $V_{ncas} = 1V$ and $V_{pcas} = 2V$ worked well. Thus, V_{ncas} needed to move 1V while V_{pcas} needed to move 3V. With equal width drains, P5 would capacitively inject three times as much charge as N7. Instead, I widened the NFET cascode transistors by a factor of three to equalize the expected feedthrough due to this capacitive gate-to-drain coupling. This worked well in simulation.

Note however that if the mismatch between the cascode transistors is greater than the inherent difference in operation between NFETs and PFETs, then mismatch could be a greater contributor to the size of the pedestal. A balance must be struck between these two factors. Simulation was used again to verify that the impact of mismatch in this regard was minor.

Consider the effects of mismatch elsewhere in the S/H circuit. Because of the feedback configuration, mismatch resulting in varying values of V_{hold} will have some effect on the range of the circuit but is generally a minor concern. Mismatch in the input pair P2 and P3 could manifest itself as an offset between V_{in} and V_{out}, however, and thus could be worrysome. The input pair (P2 and P3) and the current mirror (N8 and N9) were laid out with common-centroid and photolithographic invariance[1][5]. Their sizing was chosen to facilitate this layout.

The NMOS capacitor Ncap needs to be sized to reduce charge leakage over the hold time period. A $6\mu m \times 6\mu m$ NMOS capacitor shows acceptable leakage in simulation for temperatures up to about 50°C.

3.5.2 S/H Measured Data

The sample-and-hold performed as expected. Its input can be set to an arbitrary externally provided voltage (V_{rst} of the current-to-voltage converter, see sec. 3.4.2). The results of sweeping the input voltage V_{in} and measuring the output V_{out} are shown in fig. 3.29. The input voltage range of the sample-and-hold was designed to be 2.25–4.25V and in practice measured to be 2.45–4.42V with a 5V power supply.

This works well with the output of the preceeding circuit. Recall that the difference circuit outputs a current which is then integrated onto a capacitor and sampled by the S/H as V_{in} after a set integration time. The middle point of the range of V_{in} is set by the externally-provided reset voltage V_{rst} and can be chosen to coincide with the middle of the input range of the S/H ($3.4V$). The extent of the range of V_{in} is set by the magnitude of the output current from the difference circuit, which is a function of a bias setting and the magnitude of the difference between two sensed light intensities, and the length of the integration time. We choose an integration time during which a current corresponding to the most extreme expected intensity difference integrates up or down by $1V$, just reaching the limit of the range of the S/H.

Fig. 3.30 and fig. 3.31 show measurements of the pedestal. The pedestal is the output in hold mode minus the output in sample mode while the input is held constant.

The pedestal size is dependent on the input voltage V_{in}. For values of $V_{in} < 2.7$V, the magnitude of the pedestal is very large. Recall that V_{hold} is at a lower voltage than V_{in} and V_{out}. The difference

[1]p.59-61 of Allen and Holberg, 2002

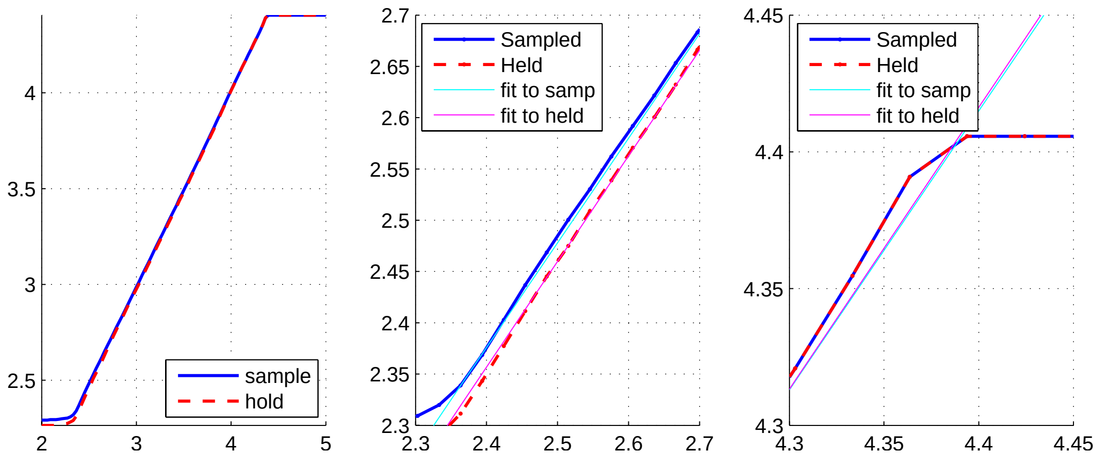


Figure 3.29: Measured data for a single sample and hold circuit. *Left:* Entire operating range. *Center:* Zoomed in near 2.5V. *Right:* Zoomed in near 4.4V. The input V_{rst} is swept (x-axis). The blue lines indicate the output of the sample-and-hold circuit in sample mode. The red line indicates the output after the transition to hold mode. As can be seen from this figure, the functional range of the circuit is from 2.45V to 4.42V.

is a function of the current in P11, since V_{hold} will move to whatever value is necessary for the currents in P10 and P11 to be equal. In the data shown, the current in P10 was large enough that for $V_{in} < 2.7$, V_{hold} dropped below the threshold voltage of Ncap. The capacitance of this MOSCAP thus decreased. For a constant amount of charge injected onto the capacitor, the resulting voltage jump (the pedestal) was thus larger. The magnitude of the pedestal for $V_{in} < 2.7V$ could be decreased by setting a lower bias current through P10, or at design time by making P10 wider.

For $2.7 < V_{in} < 3.8$, the circuit operates as previously described. Transistor N7 is in saturation and above threshold.

As V_{in} increases above 3.8V, P3 goes into subthreshold. The absolute amount of charge flowing through the channels of P5 and N7 decreases significantly. The pedestal is minimal simply because there is minimal charge that might be injected in the channels of these two transistors. In this region, most of the injected charge must be coming from gate-drain capacitive coupling as V_{ncas} and V_{pcas} swing to V_{gnd} and V_{dd}. That the pedestal is closest to zero for $V_{ncas} \approx 1.2$V when $V_{pcas} = 2.2$V suggests that at these voltages, the capacitive coupling is equal for P5 and N7.

The value of V_{ncas} also affects the size of the pedestal, as shown in fig. 3.30. At the transition from sample to hold mode, the gate of N7 drops from V_{ncas} to V_{gnd} to turn off N7. This voltage swing couples capacitively through the gate-drain capacitance of N7 onto V_{hold}, injecting negative charge onto the hold capacitor. Since for higher values of V_{ncas} this swing is larger, more negative charge is injected. For values of V_{ncas} below 0.7V, the cascode NFETs N6 and N7 turn off and the amplifier does not work (data not shown).

Similarily, the value of V_{pcas} affects the size of the pedestal as shown in fig. 3.31. As V_{pcas} is

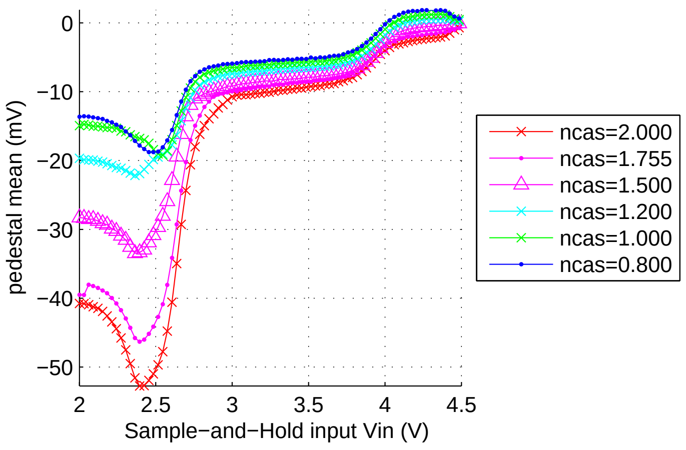


Figure 3.30: Effect of sweeping V_{ncas} on the size of the pedestal with $V_{pcas} = 2.2V$.

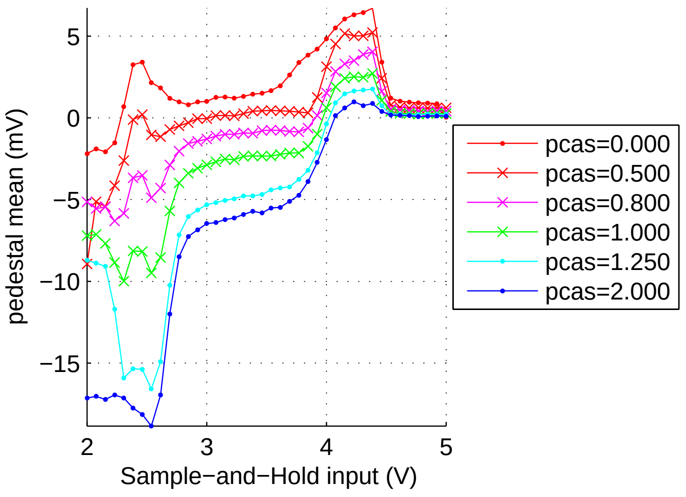


Figure 3.31: Effect of sweeping V_{pcas} on the size of the pedestal with $V_{ncas} = 1.0$V.

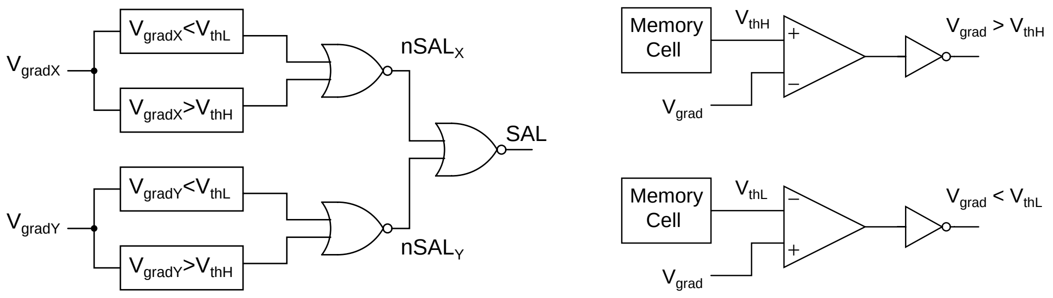


Figure 3.32: Left: The saliency circuit. If the input gradient value V_{gradX} (or V_{gradY}) exceeds the boundaries of the programmed thresholds V_{thL} and V_{thH}, the intermediate saliency value $nSAL_X$ (or $nSAL_Y$) will be zero. If both $nSAL_X$ and $nSAL_Y$ are zero, then SAL will be high. Thus, the circuit will be deemed salient only if both the X and Y gradients exceed the thresholds. Right: Subcircuits to compare the gradient values to preset thresholds. The Memory Cells [30] are programmed by tunneling and injection to store the thresholds.

increased, it must move a smaller distance to reach V_{dd} during the transition to hold mode. Less positive charge is thus injected onto V_{hold} due to capacitive coupling between the gate and drain of P5, and the pedestal becomes more negative.

3.6 Saliency Circuit

Recall the Orthogonal Gradient Detector (OGD) from sec. 2.4. To determine saliency, we wish to find all pixels which detect sufficiently large gradients in both the horizontal and vertical directions. Within each pixel, a difference circuit (described in sec. 3.4) calculates the gradients and these two values are stored in two sample-and-hold circuits (described in sec. 3.5). Thus, we have two values V_{gradX} and V_{gradY} whose values relative to V_{rst} denote the gradient value and polarity. That is, if a left-to-right gradient is denoted by $V_{gradX} > V_{rst}$, a right-to-left gradient would have $V_{gradX} < V_{rst}$. A pixel is said to be seeing a feature if the magnitudes of both gradients are suffciently large, that is:

$$(|V_{rst} - V_{gradX}| > V_{th}) \bigwedge (|V_{rst} - V_{gradY}| > V_{th}) \tag{3.11}$$

To directly implement the above equation, we need to calculate a difference, and absolute value, and a comparison. Instead, the saliency circuit is implemented equivalently by the circuit as shown in fig. 3.32. Instead of calculating an absolute value, two thresholds V_{thL} and V_{thH} are stored in floating-gate memory cells [30]. These thresholds are set to be equidistant from V_{rst}, the voltage corresponding to no gradient. We can restate the saliency constraint of eqn. 3.11 as:

$$\Big((V_{gradX} < V_{thL}) \bigvee (V_{gradX} > V_{thH})\Big) \bigwedge \Big((V_{gradY} < V_{thL}) \bigvee (V_{gradY} > V_{thH})\Big)$$

where $V_{thL} = V_{rst} - V_{th}$ and $V_{thH} = V_{rst} + V_{th}$. If the gradient values (V_{gradX} and V_{gradY}) are very close to one of the thresholds, the comparator may output a voltage midway between the rails. Driven by such an intermediate input, the gates may also output some intermediate value. To limit current draw in this scenario, the gates are drawn with moderately long transistors. The saliency output is buffered by an inverter both to increase the driving ability of the circuit and to decrease the range over which the final chip output may be at a non-binary value.

A single saliency circuit is implemented for the entire chip. As the pixels are sequentially read out, their gradient values are sequentially presented as inputs to the single saliency circuit. Thus, the binary saliency determination is made at read-out. This method requires the saliency circuit to be fast enough not to limit the read-out speed of the array. A miniumum 30Hz readout of the array and an 18×18 pixel array on this test chip calls for $100us$ readout time per pixel. Aiming to have a 1µs settling time, an approximately $1\mu A$ current driving the comparators is sufficient (in simulation). The biasing circuitry can be adjusted to provide a smaller or larger current on the fabricated chip.

Mismatch is a concern in this circuit, since we require symmetric responses for horizontal and vertical gradients. We wish each of the four comparisons to be performed identically. Note that since the threshold values for each of the four comparisons are stored on four floating gates, programming of the floating gates can compensate for offset mismatches within the four comparators.

3.7 Power Supply Sensitivity

In a practical circuit for use beyond the lab bench, effects of fluctuations in environmental variables are a concern. Floating gates remove undesired variations introduced by mismatch resulting from non-idealities of fabrication. A one-time initialization cannot compensate for changes that occur later such as fluctuations in temperature or power supply voltage. In circuits wherein calibration is performed once rather than continuously, then, we must examine the effects of these variables.

3.7.1 Effects of Power Supply Fluctuations on the Photoreceptor

Fluctuations in the photoreceptors are of little consequence for the overall task of feature detection. Since the subsequent circuit in the feature computation calculates the difference of nearby photoreceptor outputs, a common-mode increase in nearby photoreceptor outputs will be nulled in the difference computation. Power supply fluctuations resulting from perturbations external to the chip will affect nearby pixels almost identically and thus be cancelled by the difference operation. The fast clocks for the in-pixel sample-and-hold circuits switch simultaneously in all pixels, similarily producing common-mode fluctuations. The digital switches for selecting a particular pixel for readout could introduce non-uniform current draw, but since only one pixel is being read out at a time, the small total number of gates switching is unlikely to cause a notable fluctuation on the power supply. On-chip analog circuits draw relatively little current so non-uniformities in the currents drawn by the analog circuits will have negligible impact on the supply voltage. Thus, sources of power supply fluctuations on this chip either affect nearby photoreceptors in a common-mode manner that will be cancelled by subsequent computations, or are of negligible magnitude.

However, if we desire to read the photoreceptor values off chip, global fluctuations could be a concern. A full-field change in the output of the photoreceptor array may indicate a change in the overall environmental illumination or a fluctuation in the power supply, and we may care about the former but not the latter. In other applications, the photoreceptor could be embedded within a circuit that is more susceptible to local power supply fluctuations or sensitive to common-mode fluctuations of the photoreceptor output. In these contexts, we need to understand the severity and means of reducing the effects of power supply fluctuations on the photoreceptor output. The remainder of sec. 3.7.1 will discuss the pathways by which fluctuations in the power supply affect the photoreceptor output. It begins with an analytical and intuitive discussion explaining the pathways by which fluctuations in the power supply affect the photoreceptor. An understanding of the means by which the power supply affects the output suggests which nodes are most sensitive and thus where to focus to find solutions to the problem. The section concludes with simulations and measurements to substantiate the discussion and illustrate the magnitude of the effects.

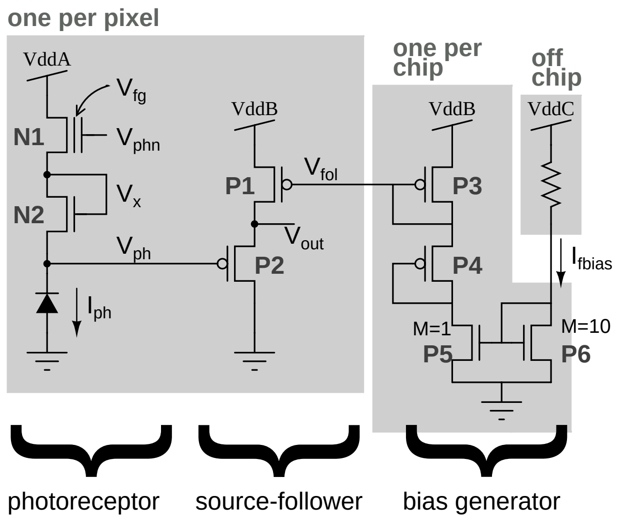


Figure 3.33: Photoreceptor and supporting circuitry. V_{fg} is the voltage on the floating gate, which allows adjustment of the DC offset of the photoreceptor output. The PFET source-follower is used within each pixel to buffer the sensitive photodiode node. An on-chip bias generator produces the voltage V_{fol} which is shared among all pixels. An externally generated bias current I_{fbias} provides a reference to the bias generator. The power supply is broken up into V_{ddA}, V_{ddB}, and V_{ddC} to facilitate discrimination of the different ways in which subcircuits are affected by their power supplies. The three supplies can be independently controlled in the fabricated chip and its test board.

The photoreceptor topology was originally introduced in fig. 3.12 and is redrawn along with some supporting circuitry in fig. 3.33. The photodiode node V_{ph} is sensitive, being driven by a potentially very small photocurrent, and thus must be buffered before being used by other circuits. Both the photoreceptor and its buffer rely on bias voltages which may be affected by power supply fluctuations, depending on how these biases are generated. We will see that while the photoreceptor output V_{ph} is fairly insensitive to fluctuations on the photoreceptor power supply V_{ddA}, it is sensitive to fluctuations in the bias voltage V_{phn}. The source-follower is sensitive to fluctuations in the value of $V_{ddB} - V_{fol}$.

First, consider the photoreceptor alone and the analytical expression characterizing it. Since the photoreceptor operates in subthreshold[2], its output can be approximated to first order by

$$V_{ph} = V_{fg} - 2 \cdot \frac{V_T}{\kappa} \cdot ln\left(\frac{I_{ph}}{I_o}\right) \tag{3.12}$$

Since $V_{ph} \propto ln(I_{ph})$, we call this a logarithmic photoreceptor. Note that V_{ddA} does not appear in this expression. This suggests that variations in V_{ddA} should have only a minor effect on the unbuffered output V_{ph}. This will be true as long as both photoreceptor transistors (N1 and N2) stay in saturation. N2 always will be saturated because it is diode-connected. We can assure that N1 remains saturated by choosing a sufficiently large floating-gate voltage V_{fg} in a mismatch-compensated photoreceptor (see fig. 3.33), or a sufficiently large bias voltage in a photoreceptor without mismatch compensation (compare fig. 3.12).

To examine the effects of power supply variation, we must consider other means by which N1 and N2 can be affected. First, the Early effect describes the increase in transistor drain current as the drain-source voltage increases ($V_{ddA} - V_x$ for N1, and $V_x - V_{ph}$ for N2). Second, the floating-gate voltage V_{fg} may be affected by other voltages that capacitively couple to the floating gate.

Including the Early effect gives the following equation to describe currents in N1 and N2:

$$I_{ph} = \underbrace{I_o e^{\kappa \frac{V_{fg}-V_x}{V_T}} \left(1 - e^{-\frac{V_{ddA}-V_x}{V_T}}\right)}_{upper\ transistor} = \underbrace{I_o e^{\kappa \frac{V_x - V_{ph}}{V_T}} \left(1 - e^{-\frac{V_x - V_{ph}}{V_T}}\right)}_{lower\ transistor} \tag{3.13}$$

where V_x is the photoreceptor node inbetween the constituent transistors. Note that V_{ddA} figures in the last term of the equation for the upper transistor. Since I_{ph} is fixed by the photocurrent,

[2]The transistor current I for a subthreshold NMOS transistor can be stated as:

$$I = I_o e^{\frac{q\kappa V_{gs}}{kT}} \left(1 - e^{-\frac{qV_{ds}}{kT}}\right) = I_o e^{\frac{\kappa V_{gs}}{V_T}} \left(1 - e^{-\frac{V_{ds}}{V_T}}\right)$$

where I_o is dependent on the process and the width-to-length ratio (W/L) of the transistor, q is the charge of an electron, k is Boltzmann's constant, T is the temperature in Kelvin, and κ parametrizes how effectively the gate couples to the channel and has a value around 0.7. V_T is the thermal voltage and equal to $\frac{kT}{q}$ or approximately 26mV. V_{ds} is the drain-to-source voltage and V_{gs} is the gate-to-source voltage. I_o and terms in the first exponential capture the first order logarithmic behavior of a subthreshold transistor. The second term, in parentheses, captures the Early effect, that is the effect of drain-to-source voltage.

a fluctuation on V_{ddA} will result in a shift in V_x. A shift in V_x will in similar fashion affect the lower transistor to cause a shift in V_{ph}. Eqn. 3.13 is difficult to solve analytically for V_{ph} but we can roughly approximate its meaning nonetheless. The transistors are operated in saturation so V_{ddA} will exceed V_x by a couple hundred millivolts. V_x exceeds V_{ph} by a similar amount. V_T is a constant equal to 26mV. Consider then the expected magnitude of the Early effect, that is, the amount by which the parenthesized term deviates from unity. Since $(V_{ddA} - V_x) \geq 200\text{mV}$, the exponent $-\frac{V_{ddA}-V_x}{V_T} \leq -7.7$, and the exponential term $e^{-\frac{V_{ddA}-V_x}{V_T}} \leq 0.000456$, so the Early effect term changes the value of the right-hand side of the equation by 0.0456% or less. Thus, we can expect fluctuations on V_{ddA} to have negligible effect on the photoreceptor output by means of the Early effect.

Consider now means by which other voltages couple capacitively to the floating gate. As stated in eqn. 3.12, the photoreceptor output is directly related to the floating-gate voltage V_{fg}. Capacitive coupling will impact a floating gate much more strongly than similar coupling would affect an actively driven node, since no mechanism opposes change due to the coupling. The voltage change on V_{fg} is directly related to the magnitude of the capacitive coupling to some secondary node and the voltage change on that node, and inversely related to the total floating-gate capacitance. This capacitive coupling is explicitly shown in fig. 3.33 between V_{phn} and V_{fg}. Other nodes including V_{ddA}, *GND*, and the tunneling voltage V_{tun} also couple to the floating gate, with the magnitude of the capacitance dependent on layout. V_{tun} couples to the poly1 floating gate via the tunneling junction, which consists of the gate oxide of a PFET whose source, drain, and well are tied to V_{tun}. V_{phn} is applied to poly2 which couples to the poly1 floating gate as a poly1-poly2 capacitor. Since gate oxide is very thin and since the poly1-poly2 oxide is also intended for making capacitors, both these capacitances should strongly couple to the floating gate. To connect the floating-gate transistor N1 with the tunneling junction, the poly1 floating gate runs over some grounded substrate and thus couples to *GND* through the field oxide. There are some unrelated bias lines which run on metal1 above the floating gate, but the much thicker oxide between poly1 and metal1 along with the small area where the two cross should result in negligible coupling to the floating gate. V_{ddA} directly couples to V_{fg} only weakly, primarily via the drain-gate capacitance of N1.

The currently fabricated chip does not allow a direct measurement of these capacitances. In simulations of the overall impact of fluctuations of the power supply on photoreceptor output, I have assumed that V_{ddA} couples to the floating gate only via V_{phn}. The close matching of my simulations to chip measurements (see fig. 3.36) suggests the validity of this assumption. On my test board, V_{phn} is generated by a resistive divider between V_{ddA} and ground. Thus, fluctuations on V_{ddA} directly change V_{phn} and thus affect V_{ph}. By the same mechanism, fluctuations of the tunneling voltage V_{tun} will affect V_{ph} (junction not shown in fig. 3.33). To minimize the capacitive coupling of external voltages onto V_{fg}, we can ground V_{phn} and V_{tun} after programming is completed. Since these two sources dominate the capacitive coupling to V_{fg}, coupling of V_{ddA} through the floating gate should be virtually eliminated.

The analysis thus far suggests that the photoreceptor is virtually impervious to power supply fluctuations. This would be true if we could operate the photoreceptor without a succeeding follower. However, the V_{ph} node is very sensitive and must be buffered. The current driving V_{ph} is only the photocurrent, which can be very small (pA or less) under indoor lighting conditions. It is thus unable to drive large loads, and the frequency response of the photoreceptor is sensitive to the capacitive load of even a small number of transistor gates. The follower used to buffer the photodiode node is the primary source of power supply sensitivity in this photoreceptor circuit. In the fabricated chip, a PFET source-follower is used to buffer V_{ph}. A PFET follower was chosen over an NFET follower for a better match with the input operating range of succeeding circuitry. A PFET follower outputs a voltage V_{out} that is roughly a diode drop *above* its input, whereas an NFET follower would output a voltage *below* its input. The voltage $V_{ddB} - V_{fol}$ sets the current through the follower. V_{out} moves to whatever voltage is necessary for P2 in the follower to source the current provided by P1, given V_{ph} on P2's gate. Neglecting the Early effect and using α to denote the ratio between the sizes of P1 and P2: $V_{ddB} - V_{fol} \approx \alpha(V_{out} - V_{ph})$. Thus, a fluctuation in the difference between V_{ddB} and V_{fol} will directly cause a fluctuation on the output V_{out}. The bias generator used on this chip was not designed with power supply rejection in mind and indeed if either V_{ddB}, I_{fbias}, or both are allowed to fluctuate, the buffered photoreceptor output is significantly affected.

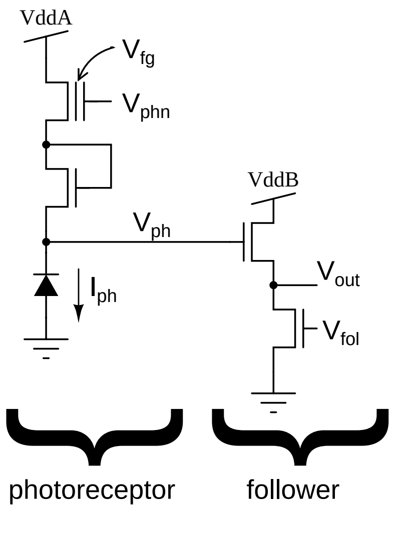


Figure 3.34: Photoreceptor with NFET follower for reduced sensitivity to power supply fluctuations (not fabricated).

In a redesign to improve power supply rejection, at least two solutions are possible. First, the bias generator could be redesigned to use a bandgap voltage reference to generate the bias V_{fol}

relative to V_{ddB}, ensuring that $V_{ddB} - V_{fol}$ stays constant independently of the power supply (and conveniently, also of temperature). Secondly, an NFET follower could be used, as shown in fig. 3.34. The NFET follower output is dependent on the difference between the bias voltage and ground and independent of V_{ddB}: $V_{fol} - V_{gnd} \approx \alpha(V_{out} - V_{ph})$. In this case, the bias voltage V_{fol} would need to be made constant relative to ground rather than the power supply. The redesign would need to accommodate the different output range of an NFET source follower.

Simulations and measurements bear out the conclusions of the above analysis. I tested several

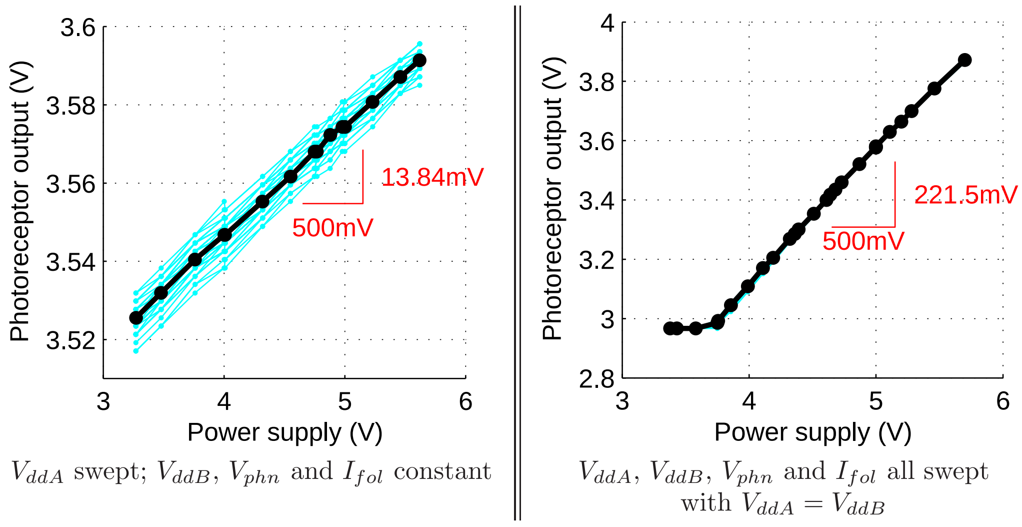


Figure 3.35: Measured data showing dependence of photoreceptor on the power supply voltage, directly on the left and including impacts on other biases on the right. Cyan lines show individual curves for each pixel in an 8x8 block. Black line and circles show mean values for those 64 pixels. The power supply can affect circuit response in several places. To separate the magnitudes of fluctuations at different points in the circuit, different biasing configurations were tried wherein only some of the supply rails and biases were swept. Shown here are the two most extreme situations. On the left, only V_{ddA} is swept. On the right, four parameters are swept simultaneously: V_{ddA} (the photoreceptor power supply), V_{ddB} (the follower power supply), V_{phn} (photoreceptor bias), and I_{fol} which controls the follower bias. Note the different scales on the y axis. The circuit topology is shown in fig. 3.33.

scenarios, varying which power supplies and biases were swept. Fig. 3.35 shows measurement data for the cases of least and greatest power supply sensitivity. Fig. 3.36 summarizes the results for all scenarios. First, note that the changes on V_{out} are linear in V_{ddA} regardless of which biasing scenario we try. Secondly, note that the effects of the two power supplies and the biases follow the trends explained previously. The photoreceptor output changes minimally as we sweep V_{ddA} only. The output is much more sensitive to fluctuations on V_{phn}, which couple to the floating gate voltage, V_{fg}, and change the operating point of the photoreceptor. The magnitude of the power supply sensitivity was greater yet when I swept biases related to the follower.

To give context to the absolute magnitudes of the voltage values, as my LCD monitor is swept from its darkest to its brightest, the resulting photoreceptor output swings over 180mV. A 13.8mV

Effect of 500mV change in V_{ddA} Measured	Simulated		Variables swept with V_{ddA}			
$V_{out}(mV)$	$V_{ph}(mV)$	$V_{out}(mV)$	V_{ddA}	V_{phn}	V_{ddB}	I_{fbias}
13.8	7.2	5.4	√			
77.0	117.7	89.4	√	√		
207.5	117.8	212.6	√	√	√	
221.7	117.8	238.8	√	√	√	√
141.0	7.3	137.8	√		√	
158.0	7.3	161.8	√		√	√

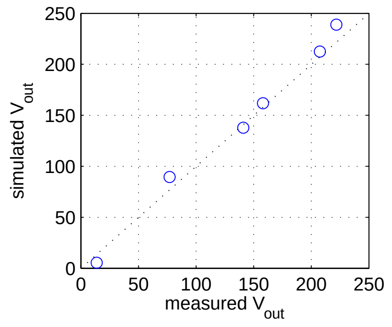


Figure 3.36: To assess the effect of power supply variation on the photoreceptor, I swept the power supply both on the fabricated chip and in simulation. The results are similar, confirming the accuracy of my simulations. I swept V_{phn} as if it were generated by a resistive divider between V_{ddA} and ground, since that was my default configuration for generating V_{phn}. That is, a 10% reduction in V_{ddA} would correspond to a 10% reduction in V_{phn} when both were swept. An on-chip bias generator creates the follower bias voltage V_{fol}. V_{fol} is swept by tying the power supply for the off-chip resistor sourcing I_{fbias} to V_{ddA}. When V_{ddB} was swept, it was tied directly to V_{ddA} and left at 5V when not swept. Note that the values listed in the table are the change on V_{out} for a 500mV change on V_{ddA}, which is much greater than the actual power supply noise that we should reasonably expect. Voltages are defined as shown in fig. 3.33.

change, as for the case of changing V_{ddA} by 500mV, represents 7.6% of the LCD monitor signal range. A 10mV fluctuation on the power supply would result in the output fluctuating over 0.15% of its range, which is negligible.

3.7.2 Effects of Power Supply Variation on Difference Circuit

Taking the lessons learned from the photoreceptor analysis, we can realize that the difference circuit (fig. 3.17)will have similar weaknesses with regards to power supply sensitivity. In the present test setup, both V_{bias} and V_{ref} are externally provided. To minimize insensitivity to power fluctuations, a bias generation circuit should reference V_{bias} to V_{dd} such that the difference $V_{dd} - V_{bias}$ stays constant.

Consider in more detail how the circuit operation is influenced by fluctuations on V_{dd}. When we move just V_{dd}, the difference between V_{dd} and V_{bias} increases, resulting in increased source-gate voltage on PX, a larger current I_γ, and larger gain. This is shown in the blue curves on the left side of fig. 3.37. This change in I_γ also affect the offset. As I_γ increases, the source voltage of P5 and P6 also increases. Since the source-drain voltage of transistors does affect their current, and since the drain voltages of P5 and P6 are not identical, the ratio of their currents may change slightly. Also,V_{dd} may couple to the floating gate fgρ and directly affect the offset-nulling ratio $I_{ref}/I_{fg\rho}$.

If we move both V_{dd} and V_{bias}, the gain mismatch is much reduced since the gate-source voltage on PX stays constant. However, the offset may still be affected as above.

Since keeping V_{bias} constant relative to V_{dd} helps tremendously with the dependence of gain on

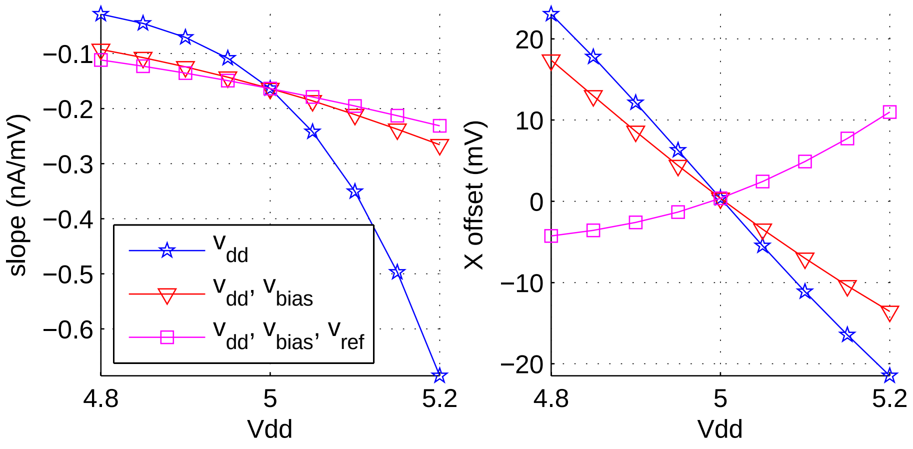


Figure 3.37: Effects of power supply fluctuations on the difference circuit (simulation). Left: The slope presented here is a measure of the change in output current I_{diff} as a function of the differential input $V_1 - V_2$. It is directly proportional to the *gain*, as introduced in fig. 3.15. Right: The offset is the input $V_1 - V_2$ necessary to produce a zero output current, also introduced in fig. 3.15. The curves in each plot correspond to sweeping a different set of biases. Since V_{bias} and V_{ref} may be generated by on-chip bias generators, their dependence on V_{dd} may vary. On the currently fabricated chip, they are provided by independent power supplies and thus independent of V_{dd} inasmuch as laboratory instruments are independent. Blue ★: Sweep of V_{dd}, with all other biases constant with respect to ground. Red triangles: Sweep of V_{dd} and V_{bias}, where $V_{dd} - V_{bias}$ is constant. Magenta squares: Sweep of V_{dd}, V_{bias}, and V_{ref}, where both $(V_{dd} - V_{bias})$ and $(V_{dd} - V_{ref})$ are constant.

power supply fluctuations, it is tempting to think that a similar trick could be employed with regards to $V_{fg\rho}$. However, the programming of $V_{fg\rho}$ is relative to V_{ref} so both values would have to be tied to V_{dd} in the same manner. A large control gate tied to V_{ref} and coupled to $V_{fg\rho}$ would reduce the differing effects of V_{dd} on these two voltages, but without active feedback, a perfect adjustment is not possible given the current topology.

3.8 Temperature Sensitivity

This section will analyze the effects of temperature variations on some of the subcircuits on this chip. I will begin with a discussion of the effects of temperature variation on individual devices: photodiodes, subthreshold transistors, and floating-gate transistors. I will then discuss in more detail temperature-induced repercussions on the photoreceptor and difference circuits.

3.8.1 Temperature Effects in the Photodiode

The photodiode is simply a reverse-biased *pn* junction fabricated as an *n*-well in the *p*-substrate. Electron-hole pairs generated within the *n*- to *p*- depletion region are swept across the depletion region by the applied electric field. Carriers generated within one diffusion length of the depletion region are likely to diffuse into the depletion region and similarily contribute to the current. Electron-hole pairs generated by photons hitting the silicon lattice result in a measurable photocurrent which is the basis for light detection in CCD and CMOS image sensors. Since the sum of the diffusion lengths in the p- and n- areas is typically larger than the width of the depletion region, photo-induced carrier generation outside the depletion region tends to contribute more to the photocurrent than carrier generation within the depletion region. Electron-hole pairs also can be thermally generated, constituting what is equivalently termed "leakage current" in a diode or "dark current" in a photodiode. Changes in the temperature of the photodiode have little effect on the photocurrent and a significant effect on the dark current. Consequently, at high illumination levels where the photocurrent is orders of magnitude greater than the dark current, temperature has negligible effect on the overall photodiode current. At low illumination levels where the photocurrent is small enough that the dark current is a significant fraction of the total diode current, increased temperatures result in increased dark current and thus a lower signal-to-noise ratio.

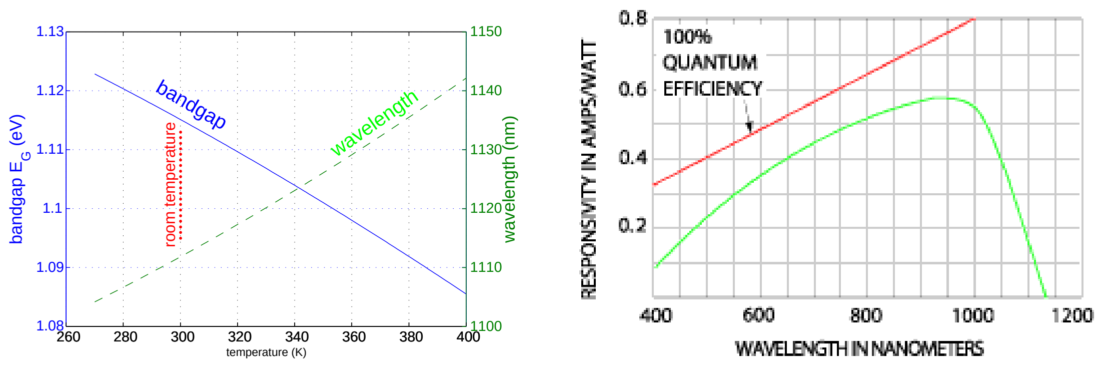


Figure 3.38: The effect of temperature on photo-generated current in the photodiode is negligible. Left: Bandgap and corresponding wavelength as a function of temperature. The blue line shows how the bandgap of silicon decreases with increasing temperature. The wavelength of photons having an energy equal to the bandgap increases proportionately, as shown by the dashed green line. Right: The efficiency with which a silicon diode generates photocurrent is a function of frequency. (Plot reprinted with permission from Melles Griot [2].) Both: From the left figure, we can see that a 10°C shift in temperature results in a 2.7nm shift in the absorption spectrum. From the right figure, we can see that this is only a tiny fraction of the absorption spectrum, which is several hundred nanometers wide.

The photo-generated current is only weakly affected by temperature. Increasing the temperature reduces the bandgap E_G as [74]:

$$E_G(T) = E_G(0) - \frac{\alpha T^2}{T + \beta}$$

where $E_G(0) = 1.16eV$, $\alpha = 7.02 \times 10^{-4} eV/K$, $\beta = 1108K$, and the temperature T is in Kelvin. Decreasing the bandgap shifts the absorption spectrum of silicon to longer wavelengths. Given the wide absorption spectrum of silicon, in practice this has a negligible effect on responsivity [2] as shown in fig. 3.38.

Dark current, arising from thermally generated electron-hole pairs, has a clear temperature dependence. This temperature dependence is somewhat complicated as it is a sum of the thermally generated diffusion currents within the n- and p- areas and the drift current within the depletion region, and the two vary differently with temperature.

Within the depletion region, the rate of generation of electron-hole pairs U is governed by [34]:

$$U \equiv -\frac{n_i}{2\tau_0}$$

where n_i is the intrinsic carrier concentration and τ_o is the effective lifetime within a reverse-biased depletion region. We shall shortly see that U varies exponentially with temperature because n_i has an exponential dependence on temperature, while τ_o is nearly independent of temperature.

The intrinsic carrier concentration n_i can be calculated from:

$$pn = n_i^2 = N_c N_v e^{-E_G/kT} \tag{3.14}$$

where E_G is the energy gap between the conduction and valence bands: $E_G = (E_c - E_v)$. N_c and N_v are the effective densities of states in the conduction and valence bands, respectively, and are each proportional to $T^{3/2}$ [74]. The intrinsic carrier concentration depends on temperature approximately in an exponential manner. The actual temperature dependence is somewhat stronger because N_c and N_v also increase with temperature.

The effective lifetime τ_o is given by:

$$\tau_o \equiv \frac{\sigma_n e^{(E_t - E_i)/kT} + \sigma_p e^{(E_i - E_t)/kT}}{2\sigma_p \sigma_n v_{th} N_t}$$

where σ_n and σ_p are the electron and hole capture cross sections, v_{th} is the carrier thermal velocity equal to $\sqrt{3kt/m*}$, N_t is the trap density, E_t is the trap energy level, and E_i is the Fermi energy level. Only the generation centers whose energy level E_t is near the Fermi level E_i contribute significantly to the generation rate, since the generation rate falls exponentially as E_t moves away from E_i. Since the centers of generation are indeed near the intrinsic Fermi level, τ_o will be nearly independent of temperature.

Carriers generated within the depletion region are almost immediately swept away by the electric field. Almost none recombine and the resulting current can be expressed as:

$$I_{dep} = q|U|WA_J = q\frac{n_i}{2\tau_0}WA_J \tag{3.15}$$

where q is the charge of an electron, W is the width of the depletion region, and A_J is the crosssectional area of the *pn* Junction. Since the rate of electron-hole generation U varies with temperature as n_i varies with temperature, the component of the thermal current generated within the depletion region I_{dep} has the same temperature dependence.

Carriers generated outside the depletion region contribute to the dark current by a somewhat different set of physical processes. Once generated, they travel by diffusion until they either recombine or diffuse to the edge of the depletion region. Only in the latter case are the carriers swept across the depletion region and thus contribute to the overall dark current. Equations describing the carriers which diffuse to the edge of the depletion region in a reverse-biased diode are [34]:

$$I_{diff,n} = qD_n \frac{n_i^2}{N_A L_n} A_J \qquad\qquad I_{diff,p} = qD_p \frac{n_i^2}{N_D L_p} A_J \tag{3.16}$$

The temperature dependence of the diffusion current is the same as that of n_i^2 [34].

In summary, the dark current has an exponential dependence on temperature. From eqns. 3.15 and 3.14 we see that the thermal current generated within the depletion region varies with temperature as $e^{-E_G/2kT}$. Eqns. 3.16 and 3.14 indicate that dark current generated outside the depletion region varies with temperature as $e^{-E_G/kT}$. Near room temperature, thermal electron-hole pair generation within the depletion region dominates the dark current and its temperature dependence is proportional to $e^{-E_G/2kT}$. Around 125°C, the two mechanisms are of similar magnitude. Above 225°C, most of the dark current is generated outside the depletion region and its magnitude changes with temperature as $e^{-E_G/kT}$.

3.8.2 Review of MOS Transistor Temperature Effects

The impacts of temperature fluctuation on a transistor are typically expressed as changes in carrier mobility μ in the channel and in threshold voltage V_t. The temperature dependence of mobility[3] can be described by [4, 74]:

$$\mu = K_\mu T^{-m} \qquad m = \frac{3}{2} \tag{3.17}$$

[3]Mobility is limited by two effects. Carrier collisions with the vibrating silicon lattice constitute lattice scattering, also known as acoustic phonons. The temperature dependence of this effect is $\mu_l \sim (m^*)^{-5/2}T^{-3/2}$ where m^* is the *conductivity effect mass*, a constant for our purposes. Carriers can also be deflected by ionic impurities with the temperature dependence being: $\mu_i \sim (m^*)^{-1/2}N_I^{-1}T^{3/2}$. The combined mobility due to both mechanisms is:

$$\mu = \left(\frac{1}{\mu_l} + \frac{1}{\mu_i}\right)^{-1} = \frac{(m^*)^{-1/2}T^{3/2}}{(m^*)^2T^3 + N_I}$$

Which effect dominates depends on the relationship between $(m^*)^2T^3$ and N_1. N_I, the ionized impurity density, depends on the doping level and also on the temperature, since ionization rates of electrons and holes decrease with increasing temperature. The material is usually doped such that the extrinsic range extends beyond the highest temperature at which the device is to be used [73]. That is to say, we choose a doping level at which lattice scattering dominates over impurity scattering and $\mu \propto T^{-3/2}$.

where $K_\mu \sim (m^*)^{-5/2}$ and is a constant, m^* is the conductivity effect mass [74] and T is temperature. The threshold voltage V_t varies with temperature as [4]:

$$V_t(T) = V_t(T_o) - \alpha(T - T_o) \tag{3.18}$$

where T_o is a reference temperature, typically 27°C, and α is approximately 2.3mV/°C. These equations are valid near room temperature [4]. Mobility shift and threshold voltage shift interact non-linearly and in opposite directions. As temperature increases, the mobility decreases, lowering the current. That same temperature increase causes decreased threshold voltage, increasing the current. The overall change in current is thus a function of both effects. Which factor dominates is a function of how deeply inverted the channel is. When the channel is strongly inverted (well above threshold), decreasing mobility becomes the more significant factor and the transistor output current decreases with increasing temperature. The opposite effect occurs below threshold: current increases with temperature. Eqns. 3.17 and 3.18 describe the measurable effects of temperature for above-threshold transistors. The effects of temperature for subthreshold transistors require further analysis.

Some texts (e.g. [4]) suggest that temperature dependence in weak inversion is dominated by voltage threshold shift. However, since the subthreshold current equation $I = I_o e^{\frac{qV_{gs}}{kT}}$ does not explicitly include a term for voltage threshold, a description of temperature effects in terms of threshold voltage shift begs the questions: "What is threshold?" and "How does its physical meaning affect the subthreshold transistor?" One way of defining threshold involves fitting a line to the output characteristics of the transistor and extrapolating. However, if we plot what happens to subthreshold voltage as we vary T in this equation, we can see that temperature shifts the slope of the line for subthreshold operation. Likewise, if we vary the mobility term $\mu \propto T^{-3/2}$ for the above-threshold equation[4] $I = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(1 - \lambda V_{ds})(V_{gs} - V_t)^2$, the slope of the curve in that regime changes. See fig. 3.39. A discussion of temperature effects that begins with threshold shift quickly becomes circular.

To understand the effects of temperature on subthreshold current, let us look at a more complete version of the subthreshold current equation [54]:

$$I = qDN_1\frac{W}{L}e^{\frac{qV_{gs}}{kT}}\left(1 - e^{-\frac{qV_{ds}}{kT}}\right) \tag{3.19}$$

[4]The above-threshold transistor current in saturation is:

$$I = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(1 - \lambda V_{ds})(V_{gs} - V_t)^2$$

and in the linear region:

$$I = \mu_n C_{ox}\frac{W}{L}V_{ds}(V_{gs} - V_t - \frac{1}{2}V_{ds})$$

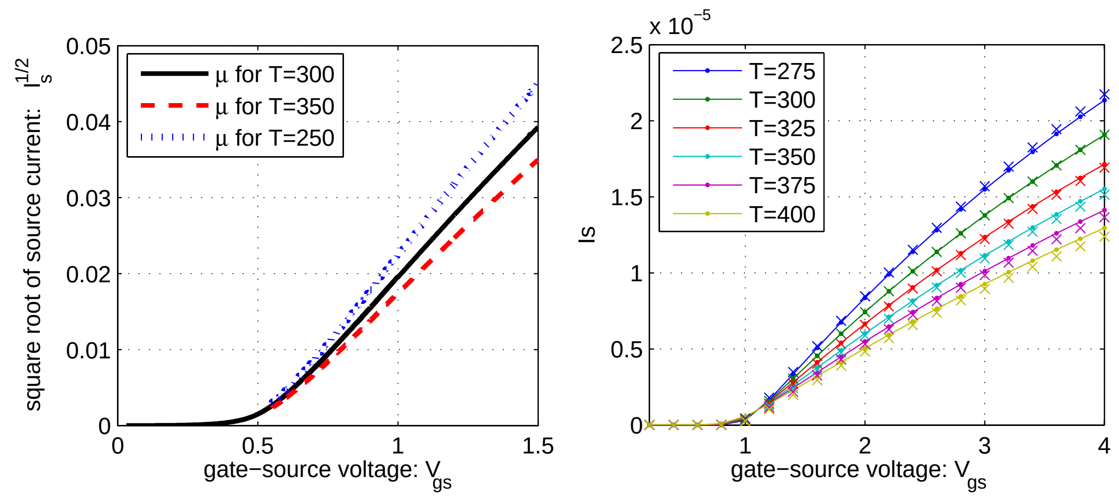


Figure 3.39: Left: Finding the threshold voltage relies on extrapolating transistor curves to find where the transition between subthreshold and above threshold operation occurs. Since these curves are shifted by temperature, discussion of how the threshold voltage shift impacts the physics of the device inherently becomes circular. Solid black curves show measured transistor current (courtesy of Jeremy Holleman). Dashed red and dotted blue curves rescale the measured data to show how it would be different if $\mu \propto T^{-3/2}$ varied in above-threshold operation. Right: The assumption that transistor current varies primarily with μ is consistent with the results of physical simulations in Atlas (courtesy of Jaideep Mavoori). Lines show results of physical simulation for several temperatures. $\times$s indicate values resulting from rescaling the green line for T=300K as if $\mu \propto T^{-3/2}$, discarding any other sources of variation with temperature.

where q is the charge of an electron, D is the diffusion constant of carriers in the channel, N_1 is the carrier density in the channel, W and L are the length and width of the device, V_{gs} is the gate-to-source voltage, and V_{ds} is the drain-to-source voltage. To observe the temperature dependencies hidden within some of these variables, expand them further. We can apply the Einstein relation to rewrite $D = \left(\frac{kT}{q}\right)\mu$. From eqn. 3.17 $\mu = K_\mu T^{-3/2}$ so that $D = \left(\frac{kT}{q}\right) K_\mu T^{-3/2}$. We can also expand N_1 in terms of the carrier density at the Fermi level N_0 and the built-in barrier between source and channel created at the time of fabrication ϕ_0: $N_1 = N_0 e^{-\phi_0/(kT)}$.

$$I = \underbrace{q\left(\frac{kT}{q}K_\mu T^{-3/2}\right)\left(N_0 e^{-\frac{\phi_0}{kT}}\right)\frac{W}{L}}_{I_o} \cdot e^{\frac{qV_{gs}}{kT}}\left(1 - e^{-\frac{qV_{ds}}{kT}}\right) \tag{3.20}$$

The above equation explicitly enumerates all dependencies on temperature for subthreshold current. This is a complicated relationship. However, we can recognize that the exponential terms involving T should dominate over $T^{-1/2}$ within the I_o term. Thus, current will vary with temperature as $I \propto e^{\frac{qV_{gs}-\phi_0}{kT}}$. For small values of V_{gs}, ϕ_0 dominates over qV_{gs} and the current increases with temperature, as reflected by the upward shifting of the lines in fig. 3.40. As V_{gs} increases, the temperature dependence of the term $e^{qV_{gs}/kT}$ increases in significance, affecting the slope of the

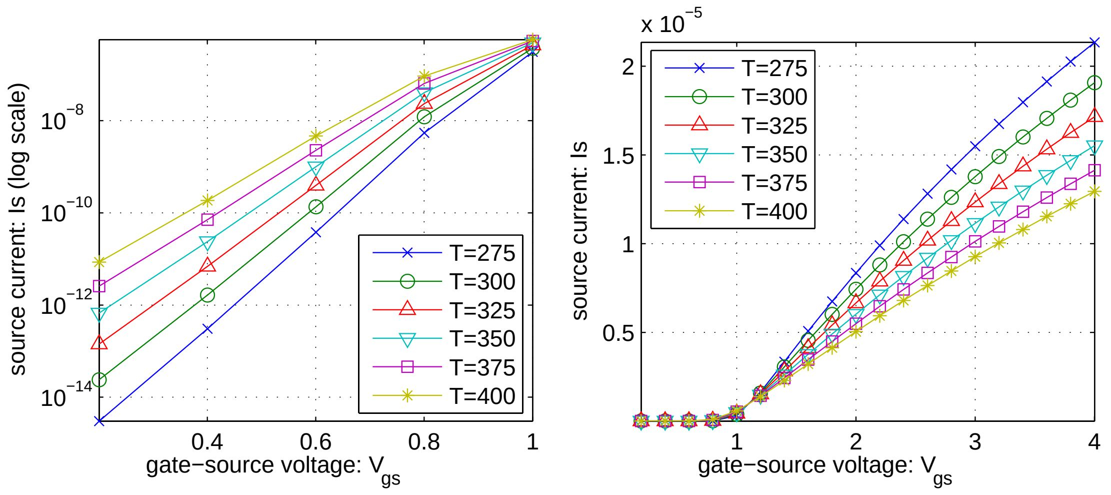


Figure 3.40: Physical simulations in Atlas show temperature dependence of transistor current. Below threshold, current increases with temperature. Above threshold, the opposite relationship holds, such that current decreases with temperature. The point where the relationship inverts is called the *temperature compensation point (TCP)*. The same data is plotted on a log scale (left) and linear scale (right). Left: Below threshold. The upward shift (on the log scale) for successively higher temperatures can be characterized by the term $e^{-\phi_0/kT}$. The decreasing slope with temperature can be explained by the term $e^{\frac{qV_{gs}}{kT}}$. Note that the simulated transistor was not saturated, so the temperature dependence of the Early effect term would decrease the current as a function of temperature by about 0.5% over the temperature range shown. Right: Above threshold.

plots. Note that the slopes decrease with temperature. The derived eqn. 3.20 is consistent with physical simulations in Atlas. Both sources indicate that the general trend in subthreshold is for current to increase with temperature, but the rate of increase lessens for large V_{gs}.

In summary, in weak inversion, increasing temperature generates electron-hole pairs in the channel. The increased number of carriers results in an increased subthreshold current. In moderate inversion, the opposite temperature coefficients of mobility and voltage threshold shift counter one another and result in small changes in channel current as a function of temperature. In strong inversion, decreasing mobility with temperature results in an overall decrease in current.

3.8.3 Impact on Floating-Gate Devices

Temperature impacts floating-gate transistors in ways identical to non-floating-gate transistors and additionally by changing the capacitance of the floating gate. Every transistor whose *poly1* gate is floating is essentially an MOS capacitor. If a temperature fluctuation causes a change in the capacitance, the floating-gate voltage will change proportionally since $Q_{fg} = C_{fg}V_{fg}$. The magnitude of the change in capacitance with temperature depends on the biasing.

The dependence of the MOS capacitance as a function of gate voltage is shown in fig. 3.41. When the gate voltage V_g is very low (below the flatband voltage V_{FB}, to be precise) and the transistor

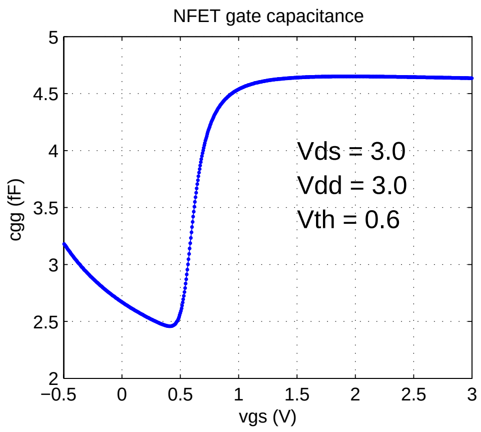


Figure 3.41: NFET gate capacitance as a function of the gate–source voltage V_{gs}, from an EKV model simulation. As described in the text, the capacitance decreases as the channel becomes more strongly depleted due to increasing gate voltage in subthreshold. When the device reaches threshold, the channel inverts and gate capacitance returns to its maximum, proportional to $\epsilon_{ox} \cdot A/t_{ox}$, where A is the area of the device.

is turned off, the capacitance per unit area between the gate and channel is simply $C_{ox} = \epsilon_{ox}/t_{ox}$, where ϵ_{ox} is the dielectric constant of SiO_2 and t_{ox} is the thickness of the oxide between the gate and channel. As V_g increases, majority carriers are pushed away and the surface of the channel becomes depleted. This can be represented as a weak downward band bending. Recall that the surface potential, ϕ_s, corresponds to the energy level $q \cdot \phi_s$ where E_C intersects the silicon-oxide interface. When $q\dot{\phi}_s$ is between E_C and E_i, the channel is depleted. Since no carriers exist in the depletion region, the effective dielectric thickness is now equal to $t_{ox} + W_{dep}$, where W_{dep} is the thickness of the depletion region, and the capacitance per unit area now equals $C_{ox} + C_{dep} = \frac{\epsilon_{ox}}{t_{ox}} + \frac{\epsilon_{si}}{W_{dep}}$. As the gate voltage increases, the bands bend more. Threshold voltage is the gate voltage at which the conduction band E_C bends enough to intersect E_F at the silicon-oxide interface (that is, $q \cdot \phi_s = E_F$) as shown in in fig. 3.42. In other words, threshold is the point at which the channel inverts. The inverted channel contains plentiful charge carriers and the dielectric thickness is again t_{ox}. Since the effective dielectric thickness in subthreshold is greater than either when the transistor is fully off or when it is fully inverted, the capacitance is lowest just below threshold.

Above threshold, the MOS capacitance depends primarily on t_{ox} and ϵ_{ox}, neither of which changes much with temperature. The temperature coefficient of the MOS capacitance is very small, on the order of 20ppm/°C [67] or 50ppm/°C [4]. In fact, the modern BSIM3V3.3 model and earlier versions of BSIM3V3 do not even include a parameter to simulate this thermal coefficient directly, and indeed simulations based on this model show no temperature dependence of the capacitance of a MOSFET. For a floating gate holding a constant charge and coupled only to above-threshold transistors, we can expect the effect of temperature on the voltage to be very small. This effect is certainly smaller

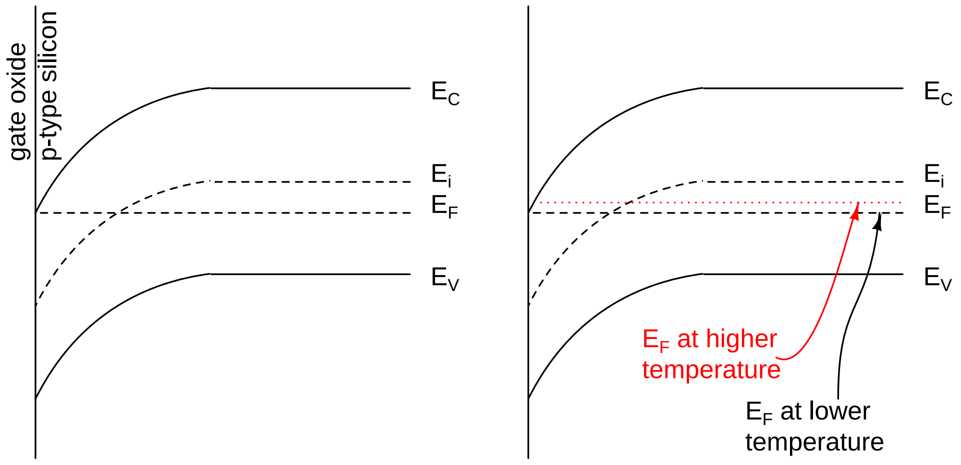


Figure 3.42: Left: The band diagram of an MIS diode. A positive bias applied on the gate causes the bands to bend downwards. When the conduction band E_c reaches the Fermi level E_F, as shown here, the device is at threshold. The potential difference between the intrinsic and Fermi levels is defined by $q\phi_b \equiv E_i - E_F$. The level of the conduction band E_C at the silicon-oxide interface defines the surface potential, ϕ_s. Right: As temperature increases, the intrinsic carrier concentration increases. This causes the Fermi level E_F to shift closer to E_i. The amount of band-bending necessary for E_C to reach E_F is thus reduced, which corresponds to a decreased threshold voltage for the device.

than the effects of mobility and threshold voltage that affect the operations of floating-gate and non-floating-gate circuits identically.

In subthreshold, on the other hand, a clear temperature dependence exists because the depletion layer thickness W_{dep} changes with temperature. W_{dep} is equal to [73]:

$$W_{dep} = \sqrt{\frac{2\epsilon_s \phi_s}{qN_a}}$$

The permittivity of silicon ϵ_s and the charge of an electron q are constants. N_a is the impurity concentration, set at fabrication. Only the surface potential ϕ_s is dependent on temperature. Calculating the value of ϕ_s is not straightforward. However, we can approach the computation from another angle. As shown in fig. 3.42, the Fermi level E_F shifts closer to E_i with temperature, equivalently expressed as a decrease in $q \cdot \phi_b$. Since threshold voltage is the voltage necessary to bend E_C enough to intersect E_F at the oxide-silcon interface, decreasing ϕ_b directly decreases the threshold voltage:

$$V_{th} = V_{FB} + 2\phi_b + \text{other terms}$$

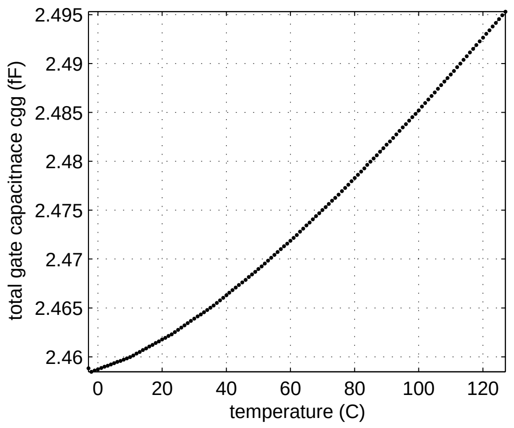


Figure 3.43: Change in gate capacitance for an n-channel transistor, as estimated from the EKV model (fig.3.41) and equations for voltage threshold shift (eqn. 3.21).

where V_{FB} is the flat-band voltage. The temperature depencence of ϕ_b is described by:

$$\phi_b = \frac{kT}{q} \cdot \ln\left(\frac{N_a}{n_i}\right)$$

where

$$n_i = 2 \cdot \left(\frac{2\pi kT}{h^2}\right)^{\frac{3}{2}} \cdot (m_n^* m_p^*)^{\frac{3}{4}} \cdot e^{-E_g/2kT}$$

and thus

$$\phi_b = \frac{kT}{q} \cdot \left[\frac{E_g}{2kT} - \frac{3}{2}\ln\left(\frac{2\pi kT}{h^2}\right) + \ln\left(\frac{N_a}{2} \cdot (m_n^* m_p^*)^{-\frac{3}{4}}\right)\right] \tag{3.21}$$

where h is Planck's constant, m_n^* and m_p^* are the effective masses of electrons and holes, and E_g is the bandgap. In short, ϕ_b will vary with temperature as $T \cdot \ln(T)$. A shift in threshold voltage is equivalent to the same magnitude shift parallel to the x-axis of fig. 3.41. Using eqn. 3.21 to compute the shift in threshold voltage and applying the resulting change in gate capacitance from the data in fig. 3.41, we estimate that for this NFET the gate capacitance will change at 2.489 fF per 100°C, or 1.18% per 100°C. The results are shown in fig. 3.43. This is an approximation and the exact change in capacitance may vary with the biasing of the device.

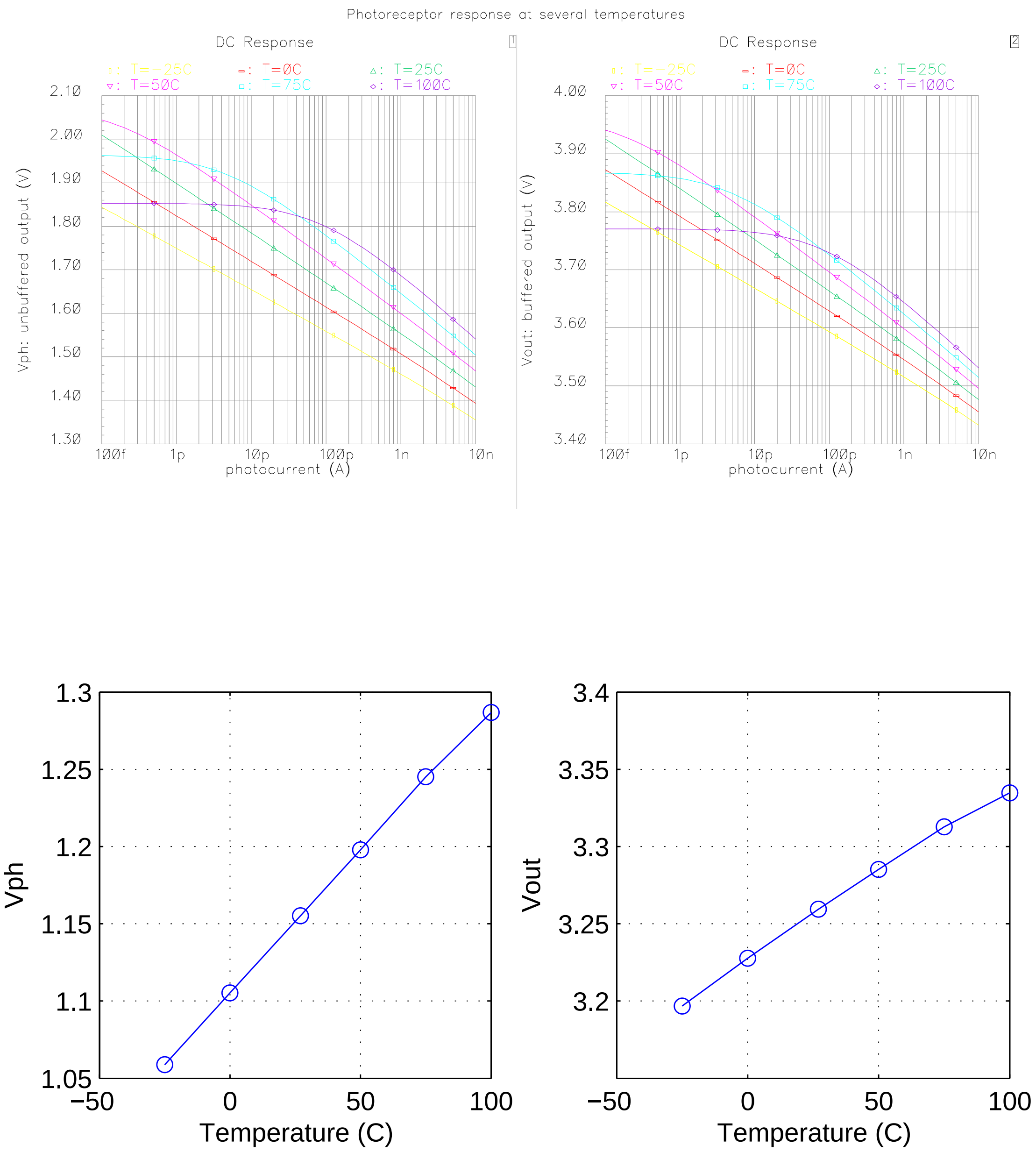


Figure 3.44: Cadence simulations of the effect of temperature on photoreceptor output, assuming constant V_{fg}. The general trend is that the photoreceptor output voltage increases by 1.835 mV/°C for V_{ph} and 1.113 mV/°C for V_{out}. Top left: Unbuffered photoreceptor output (V_{ph}) as a function of photocurrent for several temperatures. Top right: Photoreceptor output buffered by a source-follower (V_{out}) as a function of photocurrent for several temperatures. Bottom left: For a single photocurrent, unbuffered photoreceptor output as a function of temperature. Bottom right: For a single photocurrent, buffered photoreceptor output as a function of temperature.

3.8.4 Temperature Effects in the Photoreceptor

The general effect of increasing temperature within the photoreceptor will be to increase the output voltage. Recall from eqn. 3.12 the transfer function of the photoreceptor (shown in fig. 3.33):

$$V_{ph} = V_{fg} - 2 \cdot \frac{V_{TH}}{\kappa} \cdot ln\left(\frac{I_{ph}}{I_o}\right)$$

where V_{TH} is the thermal voltage $\frac{kT}{q}$. Recall from the discussion in the previous section that the term $e^{-\frac{\phi_0}{kT}}$ within I_o tends to dominate temperature effects for a subthreshold transistor. Making that term explicit by splitting I_o into two terms $I_o = I_o' \cdot e^{-\phi_0/kT}$, the photoreceptor transfer function becomes:

$$\begin{aligned} V_{ph} &= V_{fg} - 2 \cdot \frac{V_{TH}}{\kappa} \cdot ln\left(\frac{I_{ph}}{I_o' \cdot e^{-\phi_0/kT}}\right) \\ &= V_{fg} - 2 \cdot \frac{V_{TH}}{\kappa} \cdot \left(ln\left(\frac{I_{ph}}{I_o'}\right) + \frac{\phi_0}{kT}\right) \qquad (3.22) \end{aligned}$$

Thus, the photoreceptor output voltage V_{ph} should increase linearly with temperature. This is consistent with circuit simulations in Cadence, shown in fig. 3.44.

A more intuitive explanation of these equations notes that increasing temperature increases I_o roughly proportionally to $e^{-\phi_0/kT}$ for a given V_{gs} (see eqn. 3.20). In the photoreceptor, conversely, the current I_{ph} is fixed by the illumination level while V_{gs} is free to vary. As temperature increases, the transistor would like to source more current, but since the current is fixed, it must instead decrease its V_{gs}. A decrease in V_{gs} requires increasing V_{ph}. Thus, V_{ph} increases with temperature.

Another notable phenomenon in fig. 3.44 warrants explanation. For the highest temperatures and low photocurrents, the output voltage plateaus as photocurrent decreases. The photocurrent at which this plateau occurs corresponds to the sum of the substrate currents in the transistors of the photoreceptor. What is happening here is that the thermally generated current in the transistors exceeds the photocurrent, and we see the transistors responding to their own thermal current. Each of the source-bulk and drain-bulk junctions in the transistors is a reverse-biased *pn* junction. Electron-hole pairs that are thermally generated within the depletion regions of these junctions or within a diffusion length of those junctions will contribute to the thermal current (see discussion of dark current in sec. 3.8.1). The magnitude of this current is exponential in temperature and indeed we see that the interval (along the x-axis) at which the curves plateau is evenly spaced on a logarithmic scale.

The photoreceptor output V_{ph} is next buffered by a source-follower. The source-follower has a gain of less than one, which is the main reason for the apparent decreased sensitivity to temperature mentioned in fig. 3.44. Since the follower output V_{out} is almost exactly one threshold voltage above its input V_{ph}, its main temperature dependence will be to shift the output by as much as its threshold voltage changes. The resulting temperature dependence of -0.4mV/°C (simulated) is much smaller than the temperature dependence of the two subthreshold transistors constituting the photoreceptor.

Lastly, the capacitance of the floating gate will change with temperature, affecting the stored voltage on the floating gate. This effect will be additive to the effects described above. Whether a change in the floating gate capacitance increases or decreases the photoreceptor output depends on the voltages to which the floating gate is most strongly coupled, which depends on both the circuit topology and on the layout. The floating gate is intentionally coupled to a tunneling voltage V_{tun} and a control gate V_{bias}. It also couples to the channel of the injection PFET which is near ground, the channel of the photoreceptor NFET which is 1V or 2V, and parasitically to other voltages in a manner depending on layout. The total charge stored on the floating gate can be written as:

$$\begin{aligned} Q_{fg} &= C_1 \cdot (V_{fg} - V_1) + C_2 \cdot (V_{fg} - V_2) + ... + C_N \cdot (V_{fg} - V_N) \\ &= V_{fg} \cdot \left(\sum_{n=1}^{N} C_n \right) - \sum_{n=1}^{N} V_n C_n \end{aligned}$$

which can be solved for V_{fg}:

$$V_{fg} = \frac{Q + \sum_{n=1}^{N} V_n C_n}{\sum_{n=1}^{N} C_n}$$

It is clear that the magnitude and direction of the change on V_{fg} depends on the relative magnitudes of all these capacitances. For example, an increase in the coupling to a high voltage will increase V_{fg} and the photoreceptor output V_{ph} will increase by the same amount as V_{fg}. An increase in the coupling to ground will have the opposite effect and decrease V_{fg}, and thus V_{ph}. ratio of C_{vdd} to C_{gnd}. To minimize temperature dependence, we want to balance the coupling to voltages above V_{fg} and below V_{fg}. We can do this by carefully choosing the voltages to which we tie biases such as V_{tun} and V_{bias}, which are not actively used after programming and during normal operation. Because the exact values of the capacitances C_n are difficult to estimate analytically, the choice of values would be estimated best experimentally.

3.8.5 Temperature Effects in the Difference Circuit

Temperature affects the difference circuit (fig. 3.17) primarily by increasing the bias current. The transistor sourcing I_B typically operates just above threshold to provide a constant bias current to the circuit. The current through this device increases with temperature, as shown in fig. 3.46. Recall that:

$$I = \frac{1}{2}\mu_n C_{ox} \frac{W}{L} \left(V_{gs} - V_t\right)^2 \quad \text{in strong inversion}$$

$$V_t(T) = V_t(T_o) - \alpha(T - T_o) \qquad \text{from eqn. 3.18}$$

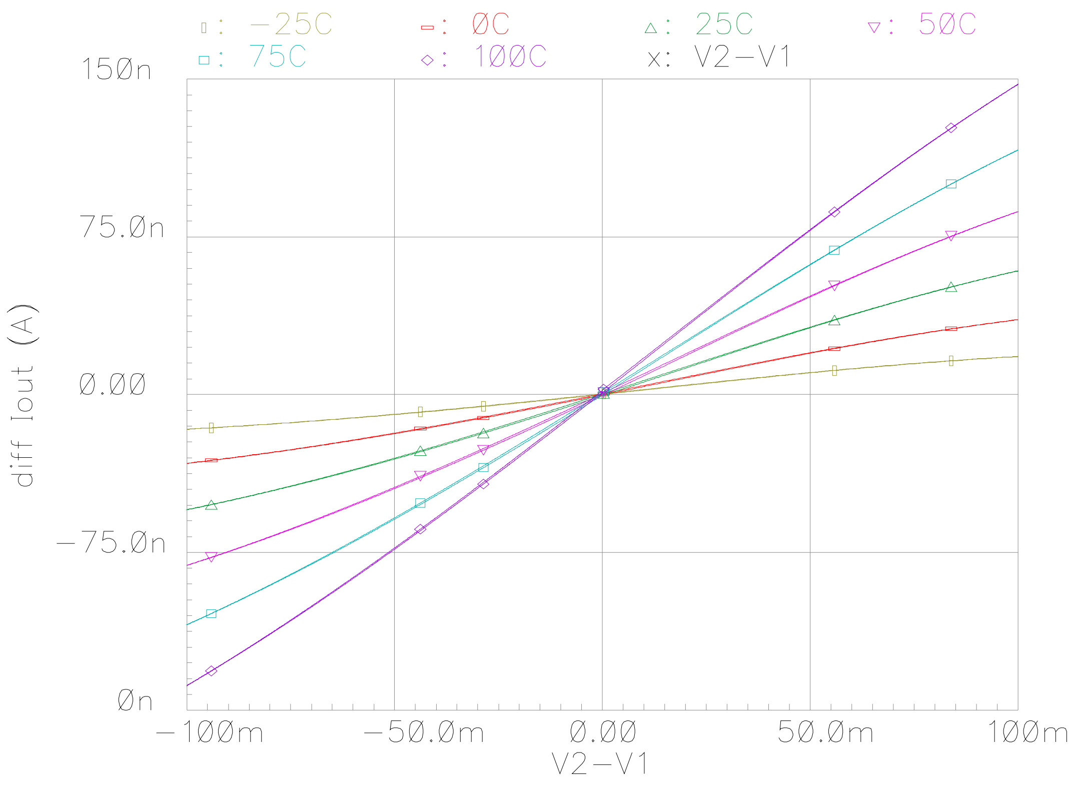


Figure 3.45: Cadence simulation of temperature effects on difference circuit.

$$\mu = K_\mu T^{-3/2} \qquad \text{from eqn. 3.17}$$

and that threshold voltage shift dominates the temperature dependence of transistors just above threshold while mobility dominates well above threshold. Since, for the simulation shown in fig. 3.46, the bias transistor operated just above threshold, its temperature dependence was dominated by a shift in V_t. Since the output current is proportional to the square of V_t, we see that the resulting bias current rises as the square of temperature.

The choice of desired bias current, and thus operating point of the bias transistor, depends on the desired and possible clocking of the sample-and-hold circuit. Recall from sec. 3.4 that the current output of this circuit is integrated onto a capacitor and then sampled by a sample-and-hold to convert from the current I_{diff} to the voltage V_{diff}. The current used to produce the shown simulation results is typical for our test setup, where the clocking speed is limited primarily by our test equipment. If a faster test setup were available, we might choose to output a larger I_{diff} and integrate it for a shorter time to generate V_{diff}. The biasing transistor PX would then operate further above threshold, the effect of mobility on temperature would begin to play a stronger role, and the overall temperature coefficient of the difference circuit would decrease in magnitude. Note that it is possible for PX to operate slightly above threshold even while the other PFETs (P1-P6) continue to operate below

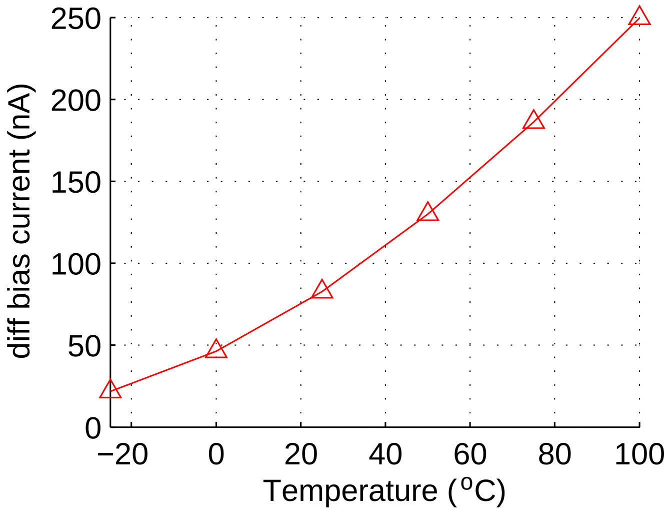


Figure 3.46: Cadence simulation of temperature effect on bias current in the difference circuit.

threshold, since the threshold voltage of PX is lower than that of the rest of the PFETs in the circuit. The wells of all the PFETs in the circuit are at V_{dd} but their sources vary. Due to the body effect, the threshold voltages of the PFETs increase as their source voltages drop. Thus, a current for which the differential pairs remain within subthreshold and compute a difference as desired may be sufficiently large to require above-threshold operation of the bias transistor. However, there is a limit of how far above threshold we can operate PX since eventually, the differential pairs in the remainder of the circuit will start coming out of subthreshold, and the computation at the core of the circuit will degrade.

The operation of the rest of the difference circuit will not be notably impacted by temperature shift. The crux of the difference computation relies on a subthreshold differential pair approximating a *tanh* function. In subthreshold transistors, temperature affects primarily the I_o term of the subthreshold current equation (eqn. 3.20). Since this factor is not dependent on the biasing of the subthreshold transistor, all subthreshold transistors will be affected in the same manner. That is, their I_o and their source currents will be multiplied by the same factor for a given temperature shift. If we look at the derivation of the *tanh* computation by a differential pair (see app. A), we note that the I_o terms for the two transistors cancel out. Since the I_o of both transistors changes with temperature by the same factor, the response of the differential pair is unaffected.

The calibration of the offset of the output is set by programming the floating gate fgρ to balance I_ρ with I_{ref} (in fig. 3.17). By similar reasoning, if V_{ref} and $V_{fg\rho}$ remain constant, this offset should be similiarily unaffected by temperature. Since the devices sourcing I_ρ and I_{ref} will have their I_o terms affected by the same temperature-dependent factor, the ratio of I_ρ to I_{ref} will not change and the calibration will remain intact. This is evidenced by Cadence simulation results shown in fig. 3.45, in that the zero-crossing of the y-axis does not change with temperature. However, as discussed in sec. 3.8.3, the gate capacitance of subthreshold MOSFETs has a temperature dependence. Given a carefully chosen and constant charge on fgρ, if the capacitance changes, the voltage will change correspondingly and the offset calibration will be disturbed. The relationship that defines what magnitude voltage corresponds to a given shift in the offset depends on the built-in offset, as discussed in sec. 3.4.1. However, the range over which we program fgρ is inevitably no greater than a couple hundred millivolts, since that is the useful range of a differential pair. Using the ballpark estimate from the end of sec. 3.8.3, if the capacitance change is 1.18% per 100°C and the range over which we might move fgρ is 200mV, and V_{ref} is around 3.5V, we might expect a change equivalent to 20% of the adjustable range for a 100°C in the temperature.

Chapter 4

Technical Conclusions

The most important things I learned in the course of my research and in the course of writing this thesis are a combination of technical knowledge directly related to electronic circuits and of ideas about project management. The technical knowledge includes topics which I have mulled over while designing my circuits, while testing them and being thankful or rueful for my design choices, and during final evaluation of the approach I had chosen. Some of these conclusions are presented in earlier parts of my thesis. Others–those which are partly a matter of opinion or which are conjectured but backed by some experience–are best discussed here, in the final part of my thesis where I may muse freely.

4.1 Engineering a Sensory System

Interacting with the complex and unpredictable world is a difficult task. Today, we laugh at statements from the 1960's when engineers and science fiction writers expected the problem to be entirely solved by the turn of the millenium.

> Machines will be capable, within twenty years, of doing any work that a man can do.
> *Herbert Simon, 1965*

As a benchmark of how far we have come in the 40 years since, consider the Defense Advanced Research Projects Agency (DARPA) Grand Challenge, created by the United States government to promote robotic development. In its first year, 2004, the best vehicle managed to travel only 7.4 miles. In its second year, 2005, five vehicles competed the 132 mile course. DARPA then created the Urban Challenge which, in 2007, will require vehicles to navigate a 60 mile mock urban area while negotiating other traffic and following traffic laws. Today's computational and algorithmic developments still are not able to replicate the skills of a nervous 16-year old in navigating the world.

Animal sensory systems vary in their complexity and processing requirements. A single-celled organism accomplishes very simple tasks guided primarily by chemical sensors. Worms perform more sophisticated behaviours, guided by touch and primitive light detection, with data processing occurring in ganglia. Mammals engage in yet more difficult tasks, relying on several sensory modalities and a hierarchy of levels of processing in their brains. Evolution developed both the sensors and the subsequent processing in small steps, starting simply. Neuromorphic engineers started at the other end, largely focusing on one of the most exciting and complex sensory modalities that requires the most processing: vision.

In retrospect, with the experience of trying to implement a system to do both sensory detection and processing, starting more simply seems like a prudent approach. It is plausible to imagine that we might be able to evaluate, engineer, implement, refine, and re-implement a simpler sensory modality, like the whisking of a rat, with only a single type of computation. Animal vision, on the other hand, tends to involve multiple types of early processing, succeeded by at least one further level of computation to merge the local responses into a global observation. The approach I took for this thesis involved incorporating a computation into every pixel. This is appropriate if a single computation is sufficient to solve a task. However, it does not scale well if multiple computations are necessary. Expanding my motion detection system to include other types of image processing would require re-engineering the chip, with each additional computation requiring the pixel to be laid out again. In short, parallel computation within an array of custom pixels makes sense for simple sensory modalities wherein a single computation may indeed suffice to complete a task, and wherein the raw sensory output is unlikely to be reused by other computations. For the task of vision, wherein multiple early processing computations can be useful, the system architecture should allow for easy incorporation of additional computations without significant modification of the existing system.

4.2 Architecture for Real-Time Processing

Designers of focal-plane processing imagers who embed computational circuitry into each pixel explain the loss of layout area as justified by gains thanks to parallel processing by every pixel. On the other end of the spectrum are those designers who read out pixel arrays and feed the data into a serial digital processor. They tout the ease of reprogramming their systems and reduced design time in comparison to custom hardware. Some argue that the lesser computational throughput of a serial system is justified by reduced chip area for the imager, thus allowing higher resolution imaging. However, rarely is the chip area of the off-the-shelf digital processor compared to the area of the analogous custom analog processing.

In retrospect, an intermediate approach seems obvious yet largely neglected. In fact, if I were designing yet another system to do image acquisition and processing based on a custom imager, I carefully would analyze this middle path. A promising compromise between the two extremes would be to combine on a single chip a dense array of simple photoreceptors with a single serial just-in-time analog processor. The cost of parallel processing is high indeed when we try to cram computation into every pixel. These analog computation blocks tend to either perform trivial operations, be poorly matched and output imprecise values, or take up impractically much area. On the other hand, if only a single block were needed to perform some analog computation, many of these problems could be ameliorated. With only one block, rather than hundreds, I could apply layout-intensive areas such as common-centroid layout to reduce mismatch problems within the circuit. Compensation with floating-gate transistors would be much faster since only one circuit would need to be calibrated. Furthermore, with only one circuit, mismatch between circuits would not even arise as a consideration. So far, this description roughly matches that of the serial digital camp. What is the benefit in using an analog computational block over a digital one, then?

The benefit comes from keeping the computation on the same chip, from reduced layout area, and from the need for fewer chips in the complete system. We simply need fewer resources. Off-chip capacitances are much greater than on-chip capacitances. It is thus easier to design fast communication into a circuit on a single die than between two chips. In short, we can present raw photoreceptor data to another on-chip subcircuit more quickly than to an identical subcircuit on another chip. For applications such as mobile robotics, a real-time system is desirable. In a serial system, this calls for the necessary computations to be performed in less time than is required to read the result for a single pixel. Consider an array of 1 million pixels that we wish to read out at 30Hz. That read-out rate allows 33ns per pixel. In a standard serial digital system, within those 33ns the system would need to digitize the photoreceptor output with an A/D and squeeze in enough clock cycles to complete the computation. In an analog system, the frequency response of some analog circuit must be high enough that the output to a step response settles within 33ns. Is it reasonable to expect this? I simulated the response to a step input to my difference circuit (presented in sec. 3.4). The settling time of this circuit, biased with a few hundred nanoamps of current, was less than 10ns and thus entirely adequate. Other circuits, such as the bump circuit [19] or a Gilbert multiplier, have a sufficiently similar structure that their settling times should be similar. Increasing the bias current would reduce settling time, if required. In short, a single analog block at the output of a chip could perform just-in-time computations in lieu of individual processing elements within every pixel.

To perform computations on small patches of the image, pixel data can be temporarily stored in on-chip sample-and-hold circuits. Some additional control circuitry would be required, either on-chip or incorporated into an external controller. Previous implementations of on-chip digital control systems exist, eg. [59], in this case for windowing. In a digital system, some equivalent sort of control circuitry must also exist, typically implemented by programming general-purpose hardware.

Consider one last question about the suggestion of on-chip just-in-time computation, as described above: Why analog, not digital? One answer is noise. With regards to feasibility, one could argue that digital computation at 30MHz (requiring a higher clock rate, if multiple steps are necessary) or analog computation with a 30ns settling time are both feasible. However, photoreceptors are sensitive. Photocurrents are small. A digital processor running at well over 30MHz will inject a lot of noise in the substrate, affecting the photodetection. A secondary consideration could be layout area. To perform the computation digitally, an on-chip A/D converter would be needed and the digital blocks would require more area than an efficient analog implementation. For example, an 8-bit digital multiplier requires 40 times more layout area than an analog version incorporating floating gates for mismatch compensation (see sec. B). For simple post-processing computations next to a dense photoreceptor array, these blocks may be just a fraction of the die area and not worth quibbling about. For more complex computations involving several multiplications, the area savings can be substantial. A commonly lauded benefit of analog over digital computation is power savings. Indeed, the power savings of an analog system can be substantial over a digital implementation, especially when subthreshold analog is employed. In many robotics applications the motors draw require so much more power than the on-board computer that the power argument for analog computations fails to stand up to reality. However, a limited number of applications does exist in which power is so limited that even processing must be considered in the energy budget, like ultra-small MEMS-based robots, medical implants, and ubiquitous computing devices intended to operate for a long time on limited battery power.

Lastly, this proposed architecture is amenable to the incorporation of multiple types of computation on the same chip, as suggested in sec. 4.1. After the raw photoreceptor data is read off from the array, it can be channeled to several on-chip analog subcircuits for parallel processing. The addition of another type of computation involves the addition on another subcircuit on the side of the array, but does not require redesigning the pixel nor the other computational blocks already in place.

In summary, I propose combining the imager and computational blocks on the same chip. I propose performing the calculations serially, since it is reasonable and feasible to make on-chip analog computations run quickly relative to off-chip read-out rates. I propose using analog computational blocks rather than digital ones, for reduced noise and efficiency in layout area.

4.3 Floating Gates for Mismatch Reduction

Floating gates are a sexy idea. We can incorporate these small structures into circuits to reduce mismatch, saving much space relative to layout techniques for mismatch correction. We can program them once and null mismatch indefinitely (or at least for years or decades). While floating-gate technology is not without its own set of design challenges, many applications could benefit from floating-gate devices to address mismatch. Floating gates are a useful tool that can enable some exciting applications when implemented well. Under what circumstances is it appropriate to incorporate floating gate transistors as a means of mismatch reduction? How difficult are these devices to program? What subtle issues must be overcome in using these devices? These types of questions, unfortunately, are rarely answered in modern journal or conference publications. In this section, I discuss little bits of wisdom and practical advice that I gleaned through the experience of working with floating-gate transistors.

4.3.1 Retention Time: Benefits and Oxide Scaling

One obvious benefit of floating-gate devices is the non-volatile nature of the charge storage. Floating gates have a long retention time made possible by storing charge on a fully-isolated conductor. In contrast, when a transistor is used as a switch to control charge flow onto a capacitor (eg. in DRAM, Dynamic Random Access Memory), subthreshold current through the transistor allows charge to leak off the capacitor. The difference in leakage rates depends on design and operational parameters, but the difference can be as large as years for floating gates versus milliseconds for traditional capacitors. Calibration is retained even if the power is cycled, unlike the latches used in SRAM (Static Random Access Memory). The obvious concern expressed by designers considering using floating-gate devices is whether they will continue to work in modern fabrication processes with their shrinking gate oxides, down to about 10Å as of time of this writing[1]. As shown in fig. 3.5, this issue is reflected in plateauing gate oxide thicknesses in processes optimized for non-volatile memories (NVM) which use floating gates. Indeed, 70-80Å is the magic number for oxide thickness below which floating-gate retention times are too short for long-term non-volatile storage. Charge leaks away due to direct tunneling across these thin oxides, at a rate exponential in the oxide thickness. However, many modern processes include as an option thicker oxide transistors having oxides of at least 70Å. The purpose of these transistors is to interface with off-chip devices that run at higher voltages (like 5V), but they can also be used for floating-gate devices. Any processes that allow for these thicker oxide transistors, then, will support floating-gate transistors with good retention times.

4.3.2 Calibration Complexity

An important consideration in designing any programmable system is how to program it. The characteristics that are unusual in floating-gate devices are their long retention times and the exponential dependence of programming rates on applied voltages. With their long retention times, floating-gate devices may be expected to be programmed once to initialize them and never again, in which case the complexity and speed of the programming algorithm introduce only an infrequent or one-time cost. Even a complicated programming procedure may not be unreasonable if it only needs to be done once. A little extra complexity in the calibration of a floating gate compared to other alternatives may be acceptable.

However, the exponential dependence on applied voltages of programming rates can necessitate complex calibration procedures, and the final system design must reflect this or else the “little extra complexity” will become an impractical hurdle. Tunneling has an exponential dependence on the voltage across the oxide. Hot carrier injection is exponentially dependent on the source-drain and gate-drain voltages of the injection transistor. Thus, slight mismatch between devices can have a dramatic impact on their relative programming rates. This can be countered in several ways, and one of these or some other technique is, in practice, essential. If properly applied, one of these techniques can make programming floating-gate devices a viable and reliable procedure.

Lastly, note that if the calibration is intended to remove offsets in a system that itself may be affected by temperature or other environmental variables, repeated calibration may be necessary.

4.3.2.1 Calibrating the Calibration

We can apply the tool to itself by using floating-gate devices to adjust some parameter that controls the injection or tunneling rate [40], first setting all calibration rates equal, and secondly calibrating the system. This is a useful technique if the system is to be recalibrated multiple times or continuously.

4.3.2.2 Continuous Adaptation

A second option is continuous adaptation [37] used with feedback. If, as is usual, a separate injection transistor shares its gate with the functional floating-gate transistor, injection can change the stored charge on the floating gate without interrupting normal operation of the remainder of the circuit. Thus, if some observable parameter is available during operation to indicate the calibration state of the circuit, the calibration can be continuously adjusted via some feedback circuit. Observable parameters may include the response to a periodic known stimulus, long-term statistics of the output, or response during a reset phase of the circuit. Continuous adaptation may be implemented within the floating-gate circuit itself or imposed by an external monitoring circuit.

It is more difficult to do continuous adaptation with tunneling. When a tunneling voltage is applied across the oxide of a tunneling junction, the capacitive coupling to the floating gate pulls the floating-gate voltage towards that of the tunneling voltage. We can reduce the magnitude of this swing by coupling a second capacitor that attempts to transiently pull the floating gate voltage down equally much as the tunneling pulse pulls it up, as in [35]. However, the accuracy to which this method succeeds is limited by the matching of the capacitances on the floating gate. In practice, the operation of the floating-gate device should be expected to change somewhat during a tunneling pulse.

4.3.2.3 Speed and External Algorithms

A third option is to use fast and sophisticated calibration controls external to the floating-gate circuit, possibly off-chip. The exponential relationship between the applied voltages and the tunneling or injection rate implies that due to slight mismatch, one structure may need an order of magnitude more programming time than another to achieve the same change in state. If we are careful to design a calibration setup that allows rapid change of tunneling and injection voltages, we can start programming each structure using conservative values and adjust these voltages in small steps until we reach a reasonable programming rate. Because a change in the floating gate voltage in turn changes the programming rate, we may need to readjust these voltages several times.

Measurement speed is critical here. One error I made in designing my chip was expecting that adaptation in my calibration algorithm would work as described above. Unfortunately for me, the input to the difference circuit is a visual stimulus, which is more difficult to change rapidly than an electrical signal without special equipment not common to an electrical engineering lab. Consequently, I spent much time writing and rewriting code to generate the visual stimulus, and the resulting algorithms were still unsatisfactorily slow. Had I thought ahead and provided an electrical way to override the inputs to the difference circuits, algorithm development and programming time both would have been vastly improved.

4.3.2.4 Constant Programming Rate

Figueroa et al. [30] developed a memory cell to address the problem of change in programming rate as the floating-gate voltage changes. Without feedback, as charge is added or removed from a floating gate, the voltage thereon changes. As already mentioned, change in floating-gate voltage affects the programming rate in a direct and exponential manner. At the cost of layout area, Figueroa et al. [30] maintain constant programming rates for a given structure by keeping the floating gate at a constant voltage, regardless of the charge stored thereon. The output of a small comparator capacitively couples to the floating gate. The comparator output swings to whatever voltage is necessary to keep the floating-gate voltage equal to some reference, and the comparator output then serves as the calibrated parameter. This memory cell is indeed as simple to use as the paper suggests.

4.3.2.5 Art and Magic

The ultimate option is sophisticated design. The best engineers are creative artists, which really means that they come up with elegant designs to address potentially critical problems. Such artful designs arise partly out of creativity, partly from willingness to take risks and try new ideas, and partly from experience and reapplying previously-seen solutions. Floating-gate circuits can sometimes be made smaller or simpler to operate through slick tricks–but an improvement in functionality in the hands of a master will happen with any tool.

4.3.3 High Voltages for Tunneling

Tunneling requires several volts across the oxide, typically more than the power supply provides. In older 2.0µm 5V processes, tunneling voltages were around 40V. In a 0.35μm 3V process, we externally apply voltages around 8V to induce tunneling. It is important to consider the best means of providing such a high voltage for programming. Two practical options include a dedicated external power supply or an on-chip charge pump. Both are viable and reasonable options.

An external power supply, to a large extent, is easier for the designer but requires more resources. The designer must situate a second power supply on the board or test bench and allocate an extra pin on the chip to provide the tunneling voltage. A bare pad should be used because a tunneling voltage can blow out the sort of protection circuitry that is typically included in a standard bonding pad. However, no further circuit design in necessary.

A charge pump on-chip requires some additional design but less off-chip support. The charge pumps I designed and tested worked as expected, so in my experience they are not particularly tricky. Note that a pin on the chip may still be necessary as a tunneling control signal, but since it switches an normal inverter, alternately it may be generated on-chip. A few external control signals can run through a demultiplexer to generate many on-chip control signals. In modern processes where the tunneling voltage may be substantially higher than the operating voltage for which the process was designed, the need for high-voltage transistors within the charge pump could complicate the design. However, avoiding the requirement of a second power supply is a vast improvement at the system level. For a stand-alone system to be used beyond a test bench, a charge pump should be seriously considered.

Chapter 5

Mentoring in Academia

Good project design, mentoring, and management are highly beneficial to the success of any substantial project, whether in industry or academia. Sadly, I have seen this truism repeatedly neglected in the halls of academia as those in charge are only indirectly rewarded for good management practices. Research laboratories vary widely in how they are run, each subject to one professor's style. Professors with an intuitive grasp of good management manage well, while those without fend as best they care to and based on anecdotal experiences acquired as graduate students. Students go to graduate school to learn and be mentored, and graduate schools rely on students to produce much of their research. Yet academic institutions select professors based on their ability to do research, not on the ability to convey that skill through teaching and mentoring, and after hiring provide little guidance on how to direct and manage students. The obvious results are that each laboratory has its own set of strengths and weaknesses which persist over years and over generations of graduate students.

More in recent years and less when I first started graduate school, I have seen programs targeted at students and intended to alleviate some of the resulting problems. Some programs aim to educate graduate students on how to manage their advisors through a direct discussion. Others promote communication among students to give the participants a baseline for reasonable expectations and appropriate responses. Yet others attempt to provide mentors to students whose official advisors are not fulfilling that role. The trend of providing increasing resources to students is a very positive improvement. Sadly, I have not heard of programs that target the more permanent component of the academic institution–the professors themselves.

One example of a program for students is a seminar at the University of Washington CSE (Computer Science and Engineering) department entitled "Managing Advisors" [56]. The title ironically points out that the role of management may befall the student, not the senior mentor. The program is run by an administrator in the department, Lindsay Michimoto. As another example, Caltech's student-run GSC (Graduate Student Council) Newsletter, sent out monthly to all graduate students, decided to run a multi-part series entitled "The Grad Student–Advisor Relationship". This is a sample quotation from the first article [9]:

> It is true some advisors only want to hear from the students when they have results, but most times they're more than willing to talk with a grad student when they are stuck.

This snippet addresses a neglect exhibited by some advisors, which in turn inspires a fear that keeps some students from interacting productively with their advisors. The writer also provides realistic encouragement. The second month's article [10], subtitled "Conflict with Your Advisor", provides a series of constructive suggestions that combine in about equal parts generally useful advice on interacting in a workplace with advice on dealing with difficult, lackadaisical, or angry people.

These anecdotal sources are indicative of broader trends that I have observed. First, many advisors are poor managers. They may be over-committed, inattentive, or fail to apply different management techniques for different people. Michimoto's seminar aims to address this problem directly. Second, the grossly imbalanced power relationship between graduate students and their advisors makes for a particularly fragile situation. In the workplace, an employee may switch managers, groups, or companies with little impact on their career. In graduate school, switching advisors tends to be tantamount to starting over. This situation amplifies the stress of any conflict, at times producing negative responses like the fear reflected in the GSC article. Third, little pressure is exerted on and few resources are provided to professors to encourage them to mentor and manage better. Hiring decisions are made based on publication record, letters of recommendation, and an invited lecture, and not based on past mentoring results. Tenure decisions likewise may emphasize research over teaching. Fourth, constructive means do exist for graduate students to avoid or alleviate problems and improve communication, and the examples above aim to facilitate such solutions.

Providing resources to help students manage their relationships with advisors is certainly a step in a good direction. Some of the skills needed to successfully manage an incognizant advisor–or recognize one's own communication weaknesses–will be useful to the student in future jobs. On the other hand, being mentored well can demonstrate good management practices by example, in addition to conveying technical expertise. Well-mentored students produce research with a clear direction and with minimal repetition of prior work.

A complete solution requires proactive involvement of both the students and the professors. The most direct impacts of widely-variable mentoring are on the students, not the professors. The most hard-hit are those students with the least ability to speak out and create change–because they are the ones with advisors who don't listen. Thus, some of the initiative for change must come from those who don't personally experience a problem. Professors are the people with the most power to insist on department-wide or university-wide changes. They are a more permanent component of the academic institution than students. A long-lasting and complete solution requires professorial involvement. Professors who mentor well should be rewarded. Professors who want to teach well but do not have the skills should be provided guidance. Professors who are poor managers and refuse to do better should be given incentives to care. Programs aimed at students, like those mentioned above, should be officially encouraged and substantially supported.

Appendix A

Derivations of Equations for Differential Pair

The differential pair is the core of the difference circuit of sec. 3.4. We derive here the equations describing its operation, after [54]. Consider the differential pair shown in fig. A.1. The currents in P1 and P2 will be:

$$I_1 = I_o e^{\frac{V_x - \kappa V_1}{V_T}} \quad \text{and} \quad I_2 = I_o e^{\frac{V_x - \kappa V_2}{V_T}} \tag{A.1}$$

where $V_T = \frac{kT}{q}$ is the thermal voltage and κ characterizes how strongly the gate of the transistor couples to the channel. Since I_B splits to form these two currents:

$$I_B = I_1 + I_2 = I_o e^{\frac{V_x}{V_T}} \left(e^{-\frac{\kappa V_1}{V_T}} + e^{-\frac{\kappa V_2}{V_T}} \right)$$

Solving for e^{V_x},

$$e^{V_x} = \frac{I_B}{I_o} \cdot \frac{1}{e^{-\frac{\kappa V_1}{V_T}} + e^{-\frac{\kappa V_2}{V_T}}} \tag{A.2}$$

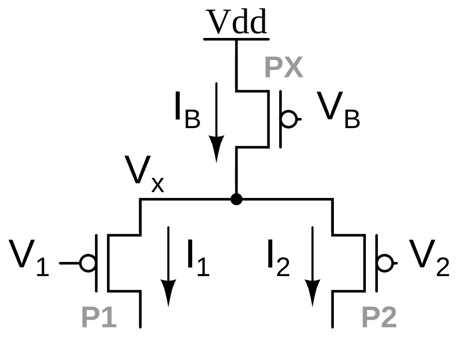


Figure A.1: A differential pair

Substituting eqn. A.2 into the expressions from eqn. A.1:

$$I_1 = I_B \frac{e^{\frac{\kappa V_1}{V_T}}}{e^{-\frac{\kappa V_1}{V_T}} + e^{-\frac{\kappa V_2}{V_T}}} \quad \text{and} \quad I_2 = I_B \frac{e^{\frac{\kappa V_2}{V_T}}}{e^{-\frac{\kappa V_1}{V_T}} + e^{-\frac{\kappa V_2}{V_T}}}$$

The difference of the currents I_1 and I_2 is:

$$I_1 - I_2 = I_B \frac{e^{\frac{\kappa V_1}{V_T}} - e^{\frac{\kappa V_2}{V_T}}}{e^{-\frac{\kappa V_1}{V_T}} + e^{-\frac{\kappa V_2}{V_T}}}$$

Multiplying the numerator and denominator by $e^{-\kappa(V_1+V_2)/2V_T}$ gives a useful expression since we can restate it using the *tanh* function:

$$\begin{aligned} I_1 - I_2 &= I_B \frac{e^{\frac{\kappa}{2V_T}(V_1-V_2)} - e^{-\frac{\kappa}{2V_T}(V_1-V_2)}}{e^{\frac{\kappa}{2V_T}(V_1-V_2)} + e^{-\frac{\kappa}{2V_T}(V_1-V_2)}} \\ &= I_B \cdot \tanh\left(\frac{\kappa(V_1 - V_2)}{2V_T}\right) \end{aligned}$$

Appendix B

Analog vs. Digital Layout Area

A digital multiplier or adder requires much more layout area and power than an analog counterpart. This section presents a rough comparison of the difference in layout area, concluding that an equivalent digital circuit is approximately 30 to 100 times larger. Analog computational circuits are limited in the number of bits of precision by noise, and 8 bits is a reasonable upper bound using typical circuit design techniques. For very precise computations with more than 8 bits, digital computation may be the only reasonable solution. For computations where 8 bits of precision suffice, such as the image data processing in this thesis, an analog computation is much cheaper in terms of layout and power.

The layout areas for my analog circuits are approximately 270μm^2 for the difference circuit and 655μm^2 for the multiplier. Comparable digital circuits are sized approximately 6600μm^2 for a ripple adder and 26400μm^2 for a multiplier. The analog circuits are 24 and 40 times smaller, respectively. Both the analog and digital layout were done in the same 0.35μm process.

The data in this section comes from two sources. Layout estimates for the analog difference circuit and analog Gilbert multiplier come from my fabricated layout. The difference circuit is presented in sec. 3.4, while the multiplier is unpublished as of the writing of this thesis. Each incorporates two floating gate devices to remove offset due to mismatch. For the digital adder and digital multiplier, I used Cadence to automatically generate layout from Verilog expressions (script modified from one by Seth Bridges [13]). Note that this is an integer multiplier, not floating-point, and that a floating-point multiplier would be substantially larger. The automatically generated layout includes all necessary gates but not the routing, which would increase the area by some amount like 10% or 20%. Cadence provides a choice of several topologies and I present results for each.

I present the data as follows. Fabricated layout areas for the analog difference circuit, the analog multiplier, and constituent subcircuits are listed in table B.1. The layout areas of digital circuits vary with the number of bits and the desired speed. Figs. B.1 and B.2 some reasonable span of these parameters, while tables B.2 and B.3 list the raw numbers used to generate the plots.

Structure	**Area**	
difference circuit excluding tun & inj	$9.3\mu m \times 20\mu m$	$186\mu m^2$
difference circuit total with tunneling junction and injection structure		$\mathbf{271\mu m^2}$
multiplier excluding tun & inj	$20\mu \times 28.5\mu m$	$570\mu m^2$
multiplier total with tunneling junction and injection structure		$\mathbf{655\mu m^2}$
tunneling junction including well spacing	$7\mu m \times 8\mu m$	$56\mu m^2$
injection structure consisting of FETs: for injection and control of injection	$5.65\mu m \times 5.10\mu m$	$29\mu m^2$
charge pump	$16.7\mu m \times 13.9\mu m$	$232\mu m^2$

Table B.1: Fabricated layout areas for a multiplier and a difference circuit in a 0.35µm process. Both use floating gate devices for mismatch compensation. The areas of the injection transistor, tunneling junction, and well spacing are included. The charge pump is not included in the totals since then number of charge pumps needed can vary from zero, to one per chip, to one per floating gate. If the tunneling voltage is provided from off-chip, no charge pumps are needed. If used, a charge pump can be shared between multiple floating-gate devices.

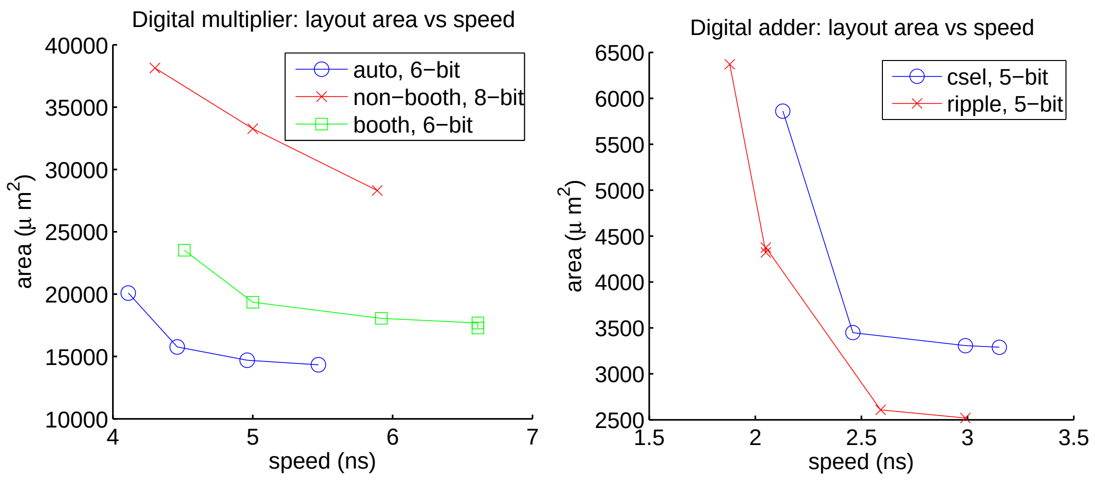


Figure B.1: Layout area (in µm^2) of a digital multiplier (left) and adder (right) as a function of speed.

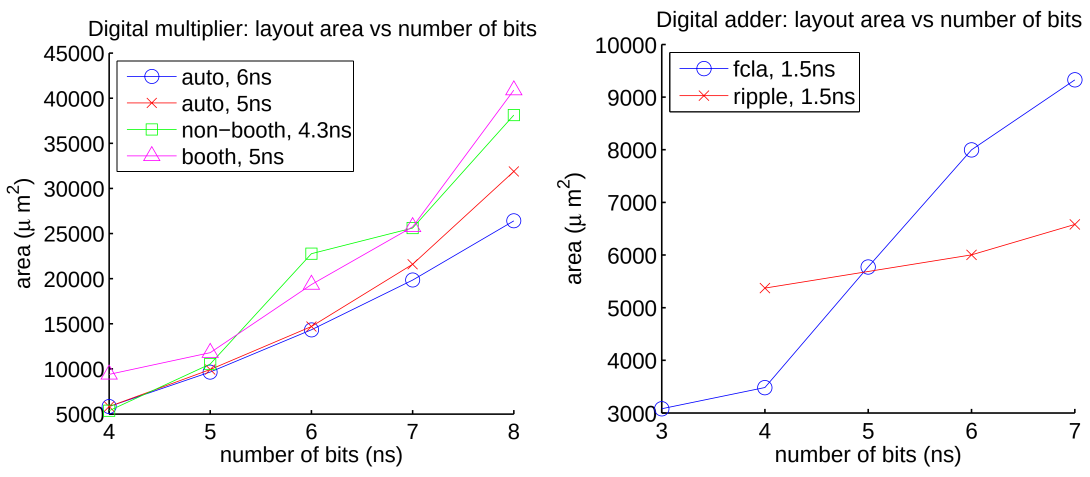


Figure B.2: Layout area (in μm^2) of a digital multiplier (left) and adder (right) as a function of the number of input bits.

topology	input	target	simulated	area	note
	bits	speed	speed		
		(ns)	(ns)	(μm^2)	
csel	5	4	3.15	3290.00	vary
csel	5	3	2.99	3307.50	speed
csel	5	2.5	2.46	3447.50	
csel	5	2.0	2.13	5862.50	
ripple	5	2.0	2.05	4322.50	compare
cla	5	2.0	1.94	3850.00	topologies
fcla	5	2.0	1.90	3517.50	
fcla	5	1.5	1.50	5775.00	
fcla	5	1.3	1.34	8715.00	
fcla	3	1.5	1.48	3080.00	vary
fcla	4	1.5	1.48	3482.50	number
fcla	6	1.5	1.50	7997.50	of bits
fcla	7	1.5	1.54	9327.50	
ripple	4	1.5	1.65	5372.50	vary
ripple	6	1.5	2.14	6002.50	number
ripple	7	1.5	2.56	6580.00	of bits
ripple	7	2.6	2.73	6107.50	vary
ripple	6	2.6	2.52	3955.00	number
ripple	5	2.6	2.59	2607.50	of bits
ripple	4	2.6	2.59	2100.00	
ripple	6	2.7	2.70	4025.00	
ripple	6	2.65	2.63	4130.00	
ripple	5	3.0	2.99	2520.00	vary
ripple	5	4.0	2.99	2520.00	speed
ripple	5	2.0	2.05	4375.00	
ripple	5	1.7	1.88	6370.00	

Table B.2: Estimated layout area for digital adders in a 0.35μm process.

topology	**input bits**	**target speed**	**simulated speed**	**area**	**note**
		(ns)	**(ns)**	**(μm^2)**	
booth	6	8.0	6.61	17692.50	
auto	6	5.0	4.96	14700.00	vary
auto	6	4.5	4.46	15767.50	speed
auto	6	4.0	4.11	20090.00	
auto	4	6.0	4.36	5827.50	
booth	4	6.0	5.22	8872.50	
non-booth	4	6.0	4.01	5372.50	
booth	6	6.0	5.92	18060.00	compare
non-booth	6	6.0	5.64	13125.00	topologies
auto	6	6.0	5.47	14332.50	at 6ns
booth	6	5.0	5.00	19355.00	compare
non-booth	6	5.0	5.00	15522.50	topologies
auto	6	5.0	4.96	14700.00	at 5ns
auto	5	5.0	4.80	9940.00	vary
auto	4	5.0	4.36	5827.50	number
auto	7	5.0	4.98	21595.00	of bits
auto	8	5.0	5.03	31885.00	at 5ns
auto	5	6.0	5.26	9660.00	vary
auto	6	6.0	5.47	14332.50	number
auto	7	6.0	5.96	19845.00	of bits
auto	8	6.0	5.88	26407.50	at 6ns
non-booth	8	6.0	5.89	28315.00	vary
non-booth	8	5.0	5.00	33267.50	speed
non-booth	8	4.3	4.30	38132.50	
non-booth	7	4.3	4.34	25602.50	vary
non-booth	6	4.3	4.37	22767.50	number
non-booth	5	4.3	4.03	10535.00	of bits
non-booth	4	4.3	4.01	5372.50	
booth	8	5.0	5.09	40897.50	vary
booth	7	5.0	4.99	25742.50	number
booth	6	5.0	5.00	19355.00	of bits
booth	5	5.0	4.96	11795.00	
booth	4	5.0	4.58	9397.50	
booth	6	4.5	4.51	23520.00	
booth	6	6.0	5.92	18060.00	vary
booth	6	7.0	6.61	17290.00	speed
booth	6	9.0	6.61	17290.00	

Table B.3: Estimated layout area for digital multipliers in a 0.35μm process.

Bibliography

[1] International Technology Roadmap for Semiconductors, 2001–2005. `http://public.itrs.net`.

[2] Understanding photodiode detector performance, Dec 2005. `http://beammeasurement.mellesgriot.com/tut_photo_det.asp`.

[3] E.H. Adelson and J.R. Bergen. Spatiotemporal energy models for the perception of motion. *J. Optical Society of America A*, 2(2):284–299, 1985.

[4] E. Allen and D. Holberg. *CMOS Analog Circuit Design*. Oxford University Press, 2002.

[5] P.E. Allen and D.R. Holberg. *CMOS Analog Circuit Design*, pages 657–662. Oxford University Press, 2nd edition, 2002.

[6] A. Aslam-Siddiqi, W. Brockherde, M. Schanz, and B.J. Hosticka. A 128-pixel CMOS image sensor with integrated analog nonvolatile memory. *IEEE J. Solid-State Circuits*, 33(10):1497–1501, 1998.

[7] M. Barbaro, P.-Y. Burgi, A. Mortara, P. Nussbaum, and F. Heitger. A 100x100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding. *IEEE J. Solid-State Circuits*, 37(2):160–172, 2002.

[8] H.B. Barlow and W.R. Levick. The mechanism of directionally selective units in rabbit's retina. *J. Physiology*, 178:477–504, 1965.

[9] O. Becker. The grad student–advisor relationship, part I: Choosing an advisor. *The GSC Newsletter*, XX(6):6–7, March 2006.

[10] O. Becker. The grad student–advisor relationship, part II: Managing conflict with your advisor. *The GSC Newsletter*, XX(7):3, April 2006.

[11] A. Benedetti and P. Perona. Real-time 2-D feature detection on a reconfigurable computer. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 593–600, 1998.

[12] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[13] S. Bridges, 2006. Personal communication.

[14] C. Chubb and G. Sperling. Drift-balanced random dot stimuli; a general basis for studying non-Fourier motion. *J. Optical Society of America*, (5):1986–2007, 1988.

[15] M. Clapp and R. Etienne-Cummings. A dual pixel-type imager for imaging and motion centroid localization. In *IEEE Int'l Symposium on Circuits and Systems*, volume 3, pages 501–504, 2001.

[16] M. Cohen and G. Cauwenberghs. Floating-gate adaptation for focal-plane online nonuniformity correction. *IEEE Trans. Circuits and Systems II–Analog and Digital Signal Processing*, 48(1):83–89, 2001.

[17] E. Culurciello, R. Etienne-Cummings, and K.A. Boahen. A biomorphic digital image sensor. *IEEE J. Solid-State Circuits*, 38(2):281–294, 2003.

[18] Liang Dai and R. Harjani. CMOS switched-op-amp-based sample-and-hold circuit. *IEEE J. Solid-State Circuits*, 35(1):109–113, 2000.

[19] T. Delbruck. Bump circuits for computing similarity and dissimilarity of analog voltages. *CNS Memo*, (26), 1993.

[20] T. Delbruck. Library essentials: analog VLSI and neural systems by Carver Mead. *The Neuromorphic Engineer Newsletter*, 1(1):11, 2004.

[21] T. Delbruck and C.A. Mead. Analog VLSI phototransduction by continous-time, adaptive, logarithmic photoreceptor circuits, 1996. Caltech Computation and Neural Systems Memo 30.

[22] F. Devos, M. Zhang, Y. Ni, and J.F. Pone. Trimming CMOS smart imager with tunnel-effect nonvolatile analog memory. *Electronics Letters*, 29(20):1766–1767, 1993.

[23] J. Diaz, E. Ros, S. Mota, F. Pelayo, and E.M. Ortigosa. Real-time optical flow computation using FPGAs. In *Early Cognitive Vision Workshop*, Isle of Skye, Scotland, UK, 2004.

[24] C. Diorio. A p-channel MOS synapse transistor with self-convergent memory writes. *IEEE Trans. Electron Devices*, 47(2):464–472, 2000.

[25] C. Diorio, P. Hasler, B.A. Minch, and C. Mead. A single-transistor silicon synapse. *IEEE Trans. Electron Devices*, 43(11):1972–1980, 1996.

[26] P.G. Drennan and C.C. McAndrew. A comprehensive MOSFET mismatch model. In *Proc. Int'l Electron Devices Meeting*, pages 167–170, 1999.

[27] R. Etienne-Cummings, J. Van der Spiegel, P. Mueller, and Mao-Zhu Zhang. A foveated silicon retina for two-dimensional tracking. *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, 47(6):504–517, 2000.

[28] C.L. Fennema and W.B. Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 9:301–315, 1979.

[29] N.J. Ferrier, S.M. Rowe, and A. Blake. Real-time traffic monitoring. In *WACV94*, pages 81–88, 1994.

[30] M. Figueroa, S. Bridges, and C. Diorio. On-chip compensation of device-mismatch effects in analog VLSI neural networks. In *Proc. Neural Information Processing Systems Conf. 17*, pages 441–448, Vancouver, Canada, 2005.

[31] P.D. Fiore, April 2002. Personal communication.

[32] P.D. Fiore, D. Kottke, W. Krawiec, and D. Gampagna. Efficient feature tracking with application to camera motion estimation. In *Conf. Record of the Thirty-Second Asilomar Conf. on Signals, Systems and Computers*, volume 2, pages 949–953, 1998.

[33] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving feature tracking with robust statistics. *Pattern Analysis and Applications*, 2:312–320, 1999.

[34] A.S. Grove. *Physics and Technology of Semiconductor Devices*. John Wiley & Sons, Inc., 1967.

[35] R.R. Harrison, J.A. Bragg, P. Hasler, B.A. Minch, and S.P. Deweerth. A CMOS programmable analog memory-cell array using floating-gate circuits. *IEEE Trans. Circuits and Systems II-Analog and Digital Signal Processing*, 48(1):4–11, 2001.

[36] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[37] P. Hasler, C. Diorio, and B.A. Minch. Continuous-time feedback in floating-gate MOS circuits. In *Proc. IEEE Int'l Symposium on Circuits and Systems (ISCAS)*, volume 3, pages 90–93, Monterey, CA, 1998.

[38] B. Hassenstein and W. Reichardt. Systemtheoretische Analyse der Zeit-, Reihenfolgen-, und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers *chlorophanus*. *Z. Naturforsch.*, (11b):513–524, 1956.

[39] D. J. Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am. A.*, (4):1455–1471, 1987.

[40] J. Holleman, 2005. Personal communication.

[41] B.K.P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[42] D. Hsu, M. Figueroa, and C. Diorio. Competitive learning with floating-gate circuits. *IEEE Trans. Neural Networks*, 13(3):732–744, 2002.

[43] Z. Hu and K. Uchimura. Motion detection from a moving observer using pure feature matching. *Int'l J. Robotics and Automation*, 15(1):21–26, 2000.

[44] A. Johnston, P.W. McOwan, and H. Buxton. A biologically plausible scheme for measuring image velocity. *J. Physiology*, (452):288, 1992.

[45] A. Johnston, P.W. McOwan, and H. Buxton. A computational model of the analysis of some first-order and second-order motion patterns by simple and complex cells. *Proc. Royal Society of London B*, (250):297–306, 1992.

[46] Z.K. Kalayjian and A.G. Andreou. Mismatch in photodiode and phototransistor arrays. In *Proc. IEEE Int'l Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 121–124, 2000.

[47] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts. A logarithmic response CMOS image sensor with on-chip calibration. *IEEE J. Solid-State Circuits*, 35(8):1146–1152, 2000.

[48] S. Kawahito, T. Eki, and Y. Tadokoro. A bit-serial column parallel processing architecture for on-sensor discrete fourier transform. In *IEEE Int'l Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 738–741, 2001.

[49] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida. A digital vision chip specialized for high-speed target tracking. *IEEE Trans. Electron Devices*, 50(1):191–199, 2003.

[50] O. Landolt, A. Mitros, and Koch C. Visual sensor with resolution enhancement by mechanical vibrations. In *Proc. 2001 Conf. Advanced Research in VLSI*, pages 249–264, Salt Lake City, Utah, 2001.

[51] M. Loose, K. Meier, and J. Schemmel. A self-calibrating single-chip CMOS camera with logarithmic response. *IEEE J. Solid-State Circuits*, 36(4):586–596, 2001.

[52] Ziyi Lu and B.E. Shi. Subpixel resolution binocular visual tracking using analog VLSI vision sensors. *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, 47(12):1468–1475, 2000.

[53] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. DARPA Image Understanding Workshop*, pages 121–130, 1984.

[54] C. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley Publishing Company, 1989.

[55] C. Michael and M. Ismail. Statistical modeling of device mismatch for analog MOS integrated circuits. *IEEE J. Solid-State Circuits*, 27(2):154–166, 1992.

[56] L. Michimoto, 2006. Post to cs-grads@cs.washington.edu, a mailing list for all graduate students in CSE department at the Univ. of Washington.

[57] T. Morita and T. Kanade. A sequential factorization method for recovering shape and motion from image streams. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(8):858–867, 1997.

[58] Y. Ni, F. Devos, M. Boujrad, and J.H. Guan. Histogram-equalization-based adaptive image sensor for real-time vision. *IEEE J. Solid-State Circuits*, 32(7):1027–1036, 1997.

[59] R.H. Nixon, S.E. Kemeny, B. Pain, C.O. Staller, and E.R. Fossum. Cmos active pixel sensor camera-on-a-chip. *IEEE J. Solid-State Circuits*, 31(12):2046–2050, Dec 1996.

[60] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers. Matching properties of MOS-transistors. *IEEE J. Solid-State Circuits*, 24(5):1433–1440, 1989.

[61] A. Pesavento. PhD thesis. Visual Sensors for Focal Plane Computation of Image Features., PhD thesis, California Institute of Technology, Pasadena, California, 2002.

[62] A. Pesavento. Personal communication.

[63] A. Pesavento and C. Koch. A wide linear range four quadrant multiplier in subthreshold CMOS. In *Proc. IEEE Int'l Symposium on Circuits and Systems*, volume 2, pages 240–243, 1999.

[64] J.M. Pimbley, M. Ghezzo, H.G. Parks, and D.M. Brown. *Advanced CMOS Process Technology.* Academic Press, Inc, 1989.

[65] C.J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997.

[66] P.O. Pouliquen, A.G. Andreou, G. Cauwenberghs, and C.W. Terrill. Learning to compensate for sensor variability at the focal plane. In *Proc. Int'l Joint Conf. on Neural Networks (IJCNN)*, volume 4, pages 2333–2336, 1999.

[67] K. Rahimi. *Adaptive-Delay Sequential Circuits.* PhD thesis, University of Washington, Seattle, Washington, 2004.

[68] L.M. Reyneri. Implementation issues of neuro-fuzzy hardware: going toward hw/sw codesign. *IEEE Trans. Neural Networks*, 14(1):176–194, 2003.

[69] R. Sarpeshkar, J. Kramer, G. Indiveri, and C. Koch. Analog VLSI architectures for motion processing: From fundamental limits to system applications. *Proc. IEEE*, 84(7):969–987, 1996.

[70] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[71] J. Segen. A camera-based system for tracking people in real time. In *Proc. 13th Int'l Conf. or Pattern Recognition*, volume 3, pages 63–67, 1996.

[72] Y. Song, L. Goncalves, and P. Perona. Learning probabilistic structure for human motion detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.

[73] B.G. Streetman. *Solid State Electronic Devices.* Prentice-Hall, Inc., 4th edition, 1995.

[74] S.M. Sze. *Physics of Semiconductor Devices.* Wiley-Interscience, 2nd edition, 1981.

[75] C. Tomasi and Kanade T. Shape and motion from image streams: a factorization method–part 3 detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[76] J.P. van Santen and Sperling G. Temporal covariance model of human motion perception. *J. Optical Society of America A*, 1(5):451–473, 1984.

[77] J.P. van Santen and Sperling G. Elaborated reichardt detectors. *J. Optical Society of America A*, 2(2):300–321, 1985.

[78] A. B. Watson and A. J. Ahumada. Model of human visual-motion sensing. *J. Optical Society of America A*, (2):322–341, 1985.

[79] M.J. Wilcox and D.C. Thelen Jr. A retina with parallel input and pulsed output, extracting high-resolution information. *IEEE Trans. Neural Networks*, 10(3):574–583, 1999.

[80] C.Y. Wu and C.F. Chiu. A new structure of the 2-D silicon retina. *IEEE J. Solid-State Circuits*, 30(8):890–897, 1995.

[81] J.L. Wyatt, Jr., D.L. Standley, and W. Yang. The MIT vision chip project: Analog VLSI systems for fast image acquisition and early vision processing. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, volume 2, pages 1330–1335, 1991.

[82] Y.-S. Yao and R. Chellappa. Tracking a dynamic set of feature points. *IEEE Tran. Image Processing*, 4(10):1382–1395, 1995.

[83] M. Zhang, F. Devos, and J.F. Pone. Trimming smart imagers for an image converter with a nonvolatile analog memory. *Sensors and Actuators*, 47(1-3):456–459, 1995.

[84] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995.

[85] Zhengyou Zhang and Ying Shan. Incremental motion estimation through local bundle adjustment, 2001. Technical Report MSR-TR-01-54.