Simulation of a Neuromorphic VLSI Optical Flow Sensor

Garrick Orchard

6/1/2009

This report describes work completed at the Institute for Neuroinformatics in Zurich Switzerland under the supervision of Giacomo Indiveri in the spring semester of 2009.

TABLE OF CONTENTS

INTRODUCTION

SCOPE

This report describes the simulation of a neuromorphic optical flow sensor as part of the project "*Implementation of biomimetic control principles using neuromorphic optic flow sensors*" at the Institute for Neuroinformatics (Zurich, Switzerland). The project abstract is repeated below for convenience.

"We propose to carry out an assessment study on "neuromorphic computation of optic flow data" by combining top expertise in custom neuromorphic VLSI motion sensor design with real-time insect flight behavior and control analysis. Specifically, we will use the unique ability of our lab to study inflight behavior of insects in real-time, for reverse engineering their ability to respond to both transversal and expanding optic flow, and combine it with our ability to design state-of-the-art neuromorphic VLSI motion sensors, for building compact low-power neuromorphic controllers. In this work we will carry out a systematic study to identify the principles used by flying insects to trigger landing responses; define the specifications for designing compact, low-power analog VLSI motion sensors; integrate the VLSI chip design and the principles derived from the fly experiments into a neuromorphic controller; and validate the neuromorphic controller to assess the potential of neuromorphic systems for space applications."

We aim to simulate landing a spacecraft by controlling thrusters with signals derived from optical flow sensors. This report is concerned with the VLSI motion sensors only. The study of the inflight behavior of insects and control principles used is beyond the scope of this report.

MOTIVATION FOR SENSING OPTICAL FLOW

Optical flow is the apparent movement within a scene due to the relative movement between the observation point and the objects within the scene. When observing a static scene, optical flow will be due purely to ego-motion, thus ego-motion can be determined from the computed optical flow.

MOTIVATION FOR NEUROMORPHIC DESIGN

Analog neuromorphic VLSI design offers an alternative approach to standard digital computation methods for sensing. Programmable digital processors have the advantage of being reconfigurable, but for specific well defined applications an application specific integrated circuit (ASIC) can be both smaller and lighter. One may argue that digital electronics are already small and lightweight, but with the target of implementation in space applications, both size and weight become important factors. VLSI allows for the photoreceptors and computation circuitry to be fabricated on a single piece of silicon and is therefore particularly well suited to optical sensing.

Digital computation is power hungry, as is the process of digitizing incoming analog signal, specifically in imaging applications when multiple pixel values must be digitised in parallel. Subthreshold analog design provides a low power alternative by removing the need for digitization and digital computation. Full chip power consumption below 1mW is typical in such designs.

The nature of analog computation also ensures that results are available almost instantaneously because computation is distributed and performed in parallel.

Designing an ASIC using neuromorphic principles can therefore result in a compact, lightweight, low-power and extremely fast sensor.

APPROACH

We approached the task of detecting optical flow by considering a recently fabricated subthreshold VLSI optical flow sensor. Using Matlab (Natick, MA), we simulate the sensor at three different levels.

1) Critical circuit blocks such as the photoreceptor are simulated at a transistor level to gain a better understanding of their operation and to predict their response to known signals. These simulations are typically run with a microsecond time-step and span only a few milliseconds.

2) A low level full chip simulation which simulates each part of the chip using approximate equations is used to verify the operation of the chip and investigate the sensitivity to transistor mismatch. The optical inputs for the low level simulations are extracted from images generated by Pangu, a simulator capable of generating images of a planet surface. These simulations are also typically run with a microsecond timestep, but the entire chip can be simulated for up to half a second.

3) A high level simulation of the chip was written based on the low level simulations. This high level simulation is used to provide approximate sensor outputs for use in the planetary landing simulations. The timestep for this simulation is arbitrary, the sensor simply provides an output when queried, but the simulation can span minutes or hours.

CHIP DESCRIPTION

The sensor detects optical flow along the axis of a single 64 pixel linear array of photoreceptors. The chip also detects the temporal derivative and spatial derivative at each pixel. The chip has a built in programmable bias generator which allows it to be easily reconfigured for performance under different operating conditions. A brief description of the operation of the chip is provided on the following pages.

BLOCK DIAGRAM OVERVIEW OF CHIP OPERATION

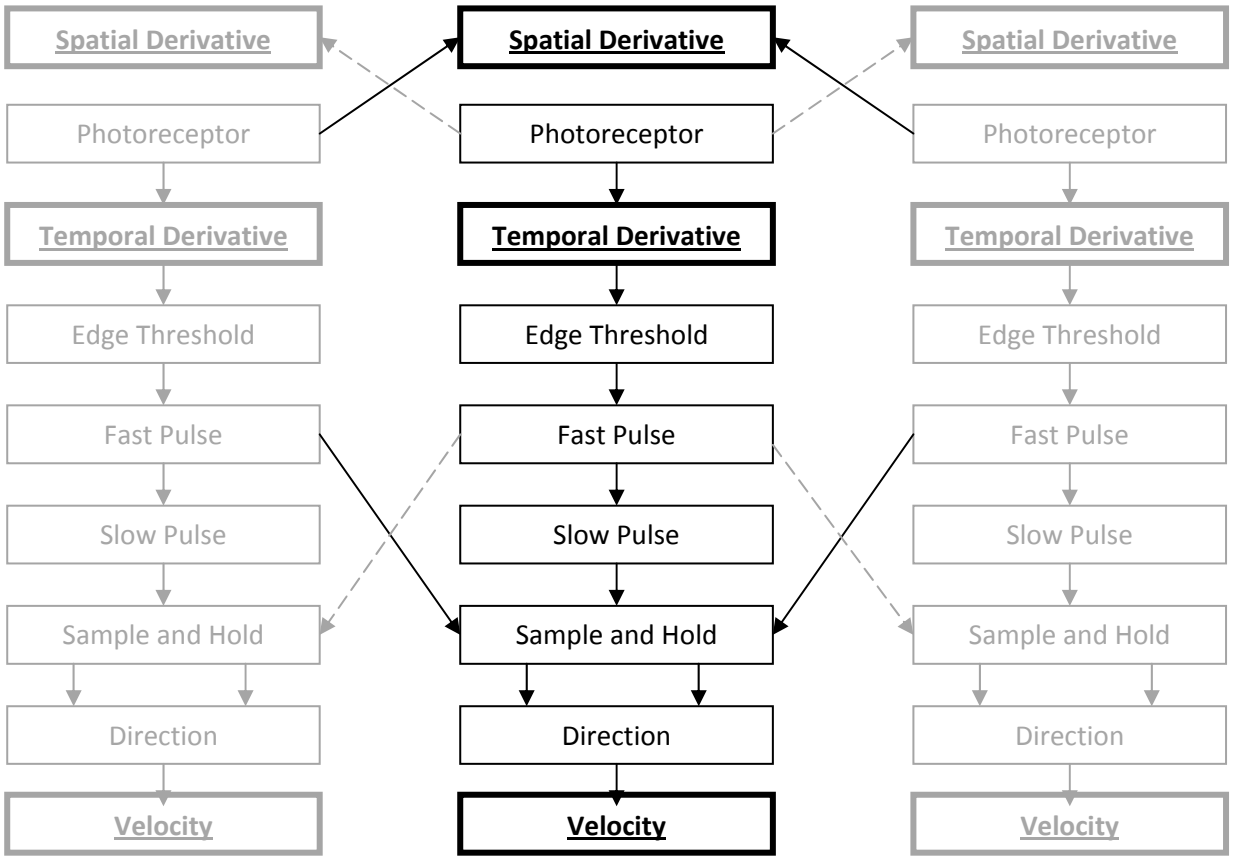

Figure 1: Block diagram showing three adjacent pixels and the information flow leading to measurements of spatial derivative, temporal derivative and velocity.

The blocks in fig 1 are described in detail in later sections, but the basic operation is as follows.

1: The spatial derivative is approximated as simply the difference between the outputs of the photoreceptors to the left and right.

2: The temporal derivative of the photoreceptor output is obtained using a high pass active filter.

3: When a sharp enough edge passes over the pixel, the temporal derivative will pass a threshold and the 'Edge Threshold' block will output a current which triggers the 'Fast Pulse' block to generate a fixed length digital pulse. This triggers the 'Slow Pulse' block to generate decaying waveform (with a known decay). The waveform is sampled in the 'Sample and Hold' block when the pixel to the right or left triggers a digital pulse, thus creating two samples. The direction block then decides which of these two samples is most relevant (recent) by determining which is larger. The larger sample is then output as a measurement of the velocity.

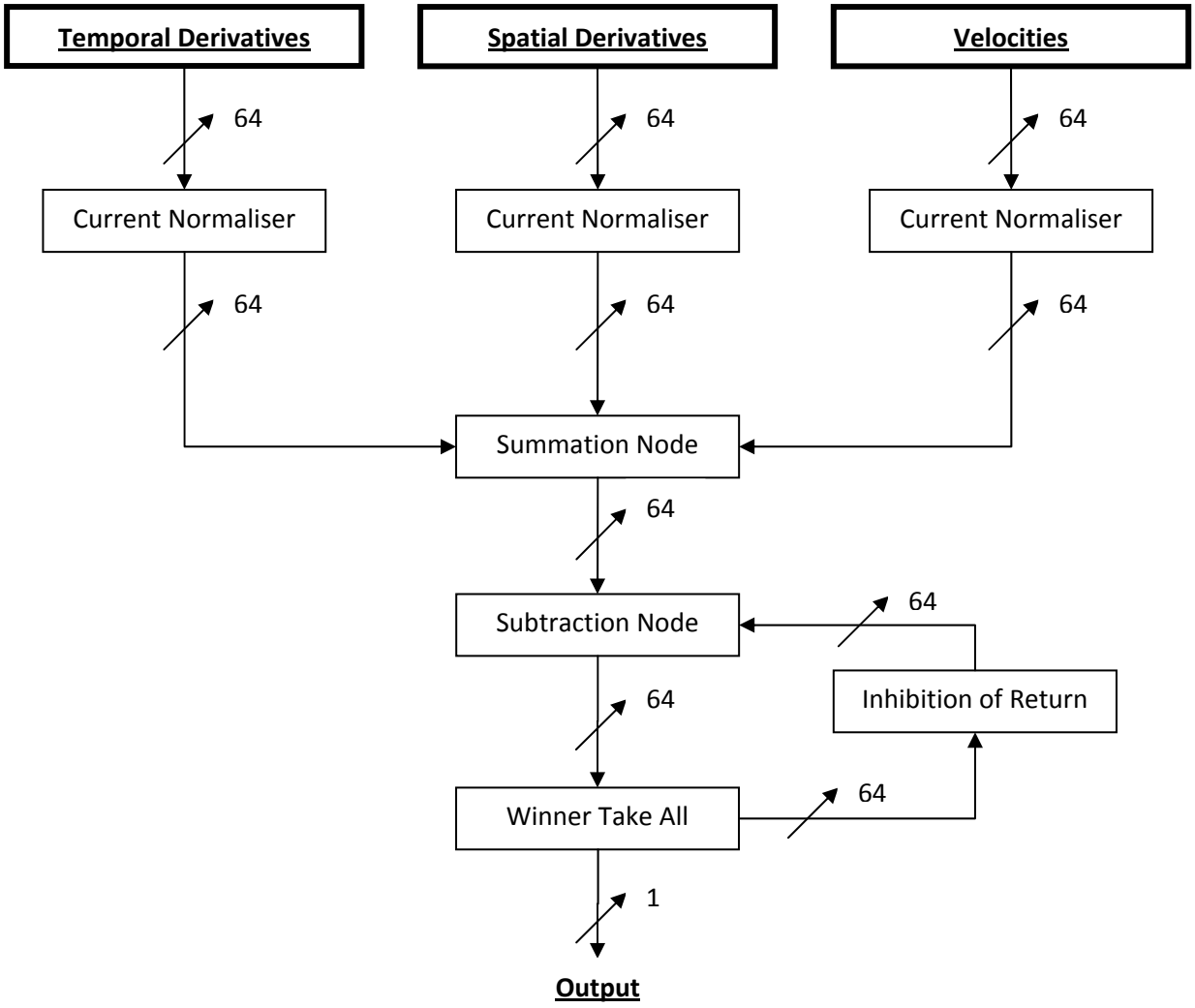

Figure 2: Block diagram showing how the pixel output to be read is determined.

Fig 2 above shows the method of determining which pixel's output should be considered. The spatial derivatives, temporal derivative and velocities are all normalised and summed before being passed into the winner take all, which determines which of the inputs is the greatest. The current normalisers are configurable via the bias current, allowing the relative weighting of the spatial derivative, temporal derivative and velocity to be controlled. In this manner we can choose to configure the sensor to only consider the strongest contrast point in the image or the fastest moving point in the image or a combination of the two.

The winner take all has hysteresis to prevent rapid jumping between pixels in the presence of noise. Lateral excitation is also implemented to ensure a smooth transition from one pixel to the next as an edge propagates across the array.

The 'Inhibition of Return' block acts to inhibit the current winner of the winner take all. The degree of inhibition increases over time, thus further encouraging the winner take all to select a new winner as time goes on. This acts to prevent the winner take all from getting 'stuck' on a single pixel and further encourages transition from one pixel to another.

TRANSISTOR LEVEL SIMULATION

The photoreceptor, temporal derivative, slow pulse and inhibition of return blocks were simulated using transistor current/voltage equations. Differential equations for each circuit were derived and solved using the Matlab 'ODE15S' function.

The slow pulse and inhibition of return blocks are consist of the same circuitry just with different transistor sizes and capacitor values. A discussion of the transistor level simulations for each block follows.

PHOTORECEPTOR

The photoreceptor [3] is designed to detect only changes in brightness. The feedback loop ensures that low frequencies are removed by adapting the operating point of the photoreceptor to new lighting conditions.

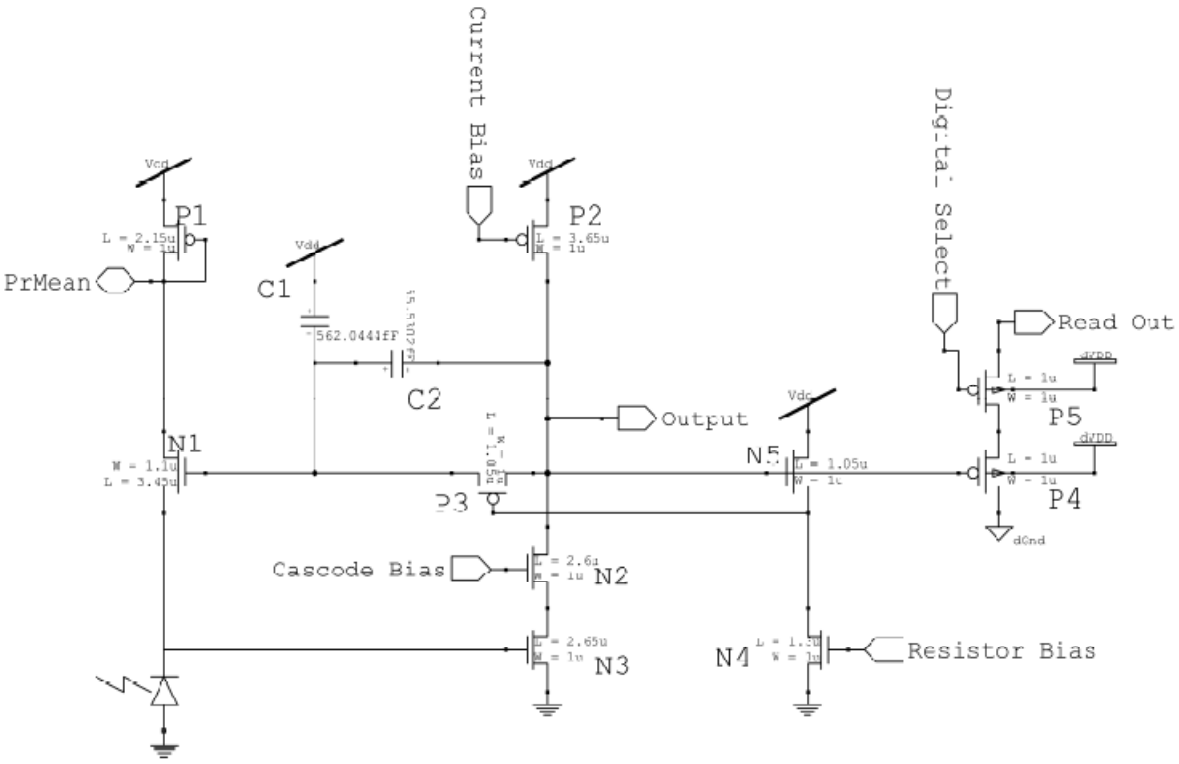

Figure 3: Photoreceptor schematic. V_r is the voltage on the photodiode, V_f is the gate of N1 and V_{fbres} is the gate of P3

The node *'PrMean'* is common to all photoreceptors, thus ensuring that P1 for each photoreceptor generates the same current, ensuring the same operating point for each photodiode. P3 acts as a feedback resistor with a value which can be controlled using *'Resistor Bias'*. P4 and P5 are only required for readout circuitry.

SIMULATION

CIRCUIT PARAMTERS

The photodiode was modelled as a capacitance in parallel with a current source which acts as the input signal. The capacitance can be approximated from the layout as

$$C_{diode} = WLC_{Sub} + 2(W + L)C_{jsw}$$

Where 'W' and 'L' are the width and length respectively of the photodiode.

$$W = 16\mu m$$

$$L = 16\mu m$$

$$C_{sub} = 0.08\frac{fF}{\mu m^2}$$

$$C_{jsw} = 0.51\frac{fF}{\mu m}$$

CIRCUIT CURRENTS

First the gate voltage of the transistor in the feedback loop is given by

$$V_{fbres} = V_{out} - V_{Resistor\ Bias}$$

Using the gate voltage of the feedback "resistor" transistor, the current will be

$$I_{feedback} = f(x) = \begin{cases} \left(\frac{W}{L}\right)_{P3} I_{0P} e^{-\frac{\kappa_p(V_{fbres}-3.3)}{U_t}} \left(e^{\frac{V_{out}-3.3}{U_t}} - e^{\frac{V_f-3.3}{Ut}}\right), & V_{out} > V_f \\ -\left(\frac{W}{L}\right)_{P3} I_{0P} e^{-\frac{\kappa_p(V_{fbres}-3.3)}{U_t}} \left(e^{\frac{V_f-3.3}{U_t}} - e^{\frac{V_{out}-3.3}{Ut}}\right), & V_f \geq V_{out} \end{cases}$$

The current through N1 is then determined in a similar manner:

$$I_{N1} = \left(\frac{W}{L}\right)_{N1} I_{0N} e^{\frac{\kappa_n V_f}{U_t}} \left(e^{\frac{-V_r}{U_t}} - e^{\frac{-3.3}{Ut}}\right)$$

Finally, the current through N3 is given by

$$I_{N3} = \left(\frac{W}{L}\right)_{N3} I_{0N} e^{\frac{\kappa_n V_r}{U_t}}$$

DIFFERENTIAL EQUATIONS

Knowing the circuit currents and capacitances we can approximate the derivative of the voltages at each time-step as:

$$dV_r = \frac{I_{N1} - I_{diode}}{C_{diode}}$$

$$dV_{out} = \frac{(I_{bias} - I_{N3})}{\left(\frac{C_1 C_2}{C_1 + C_2}\right)}$$

$$dV_f = dV_{out} \frac{C_2}{C_1 + C_2} + \frac{I_{feedback}}{C_1 + C_2}$$

RESULTS

Simulations show an asymmetrical adaptation. The adaptation after negative edges is much faster than after positive edges. This sometimes causes the circuit to over-adapt as seen in the examples below. These behaviours were also seen in measurements made on the actual chip when focused on a flashing LED. The simulations below are generated with a square wave input to the photoreceptor.

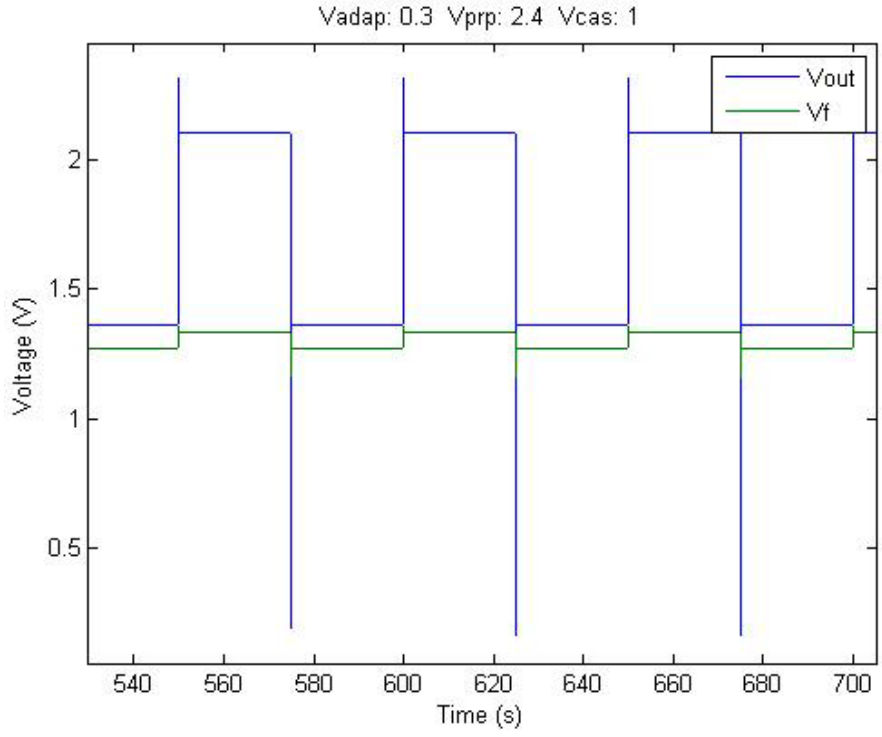

Figure 4: In this first case the adaptation bias is very low and therefore adaptation is not noticeable. It is interesting to note that on negative edges, the output briefly goes below the feedback voltage, but then the circuit overcompensates, immediately pulling it back above the feedback voltage.

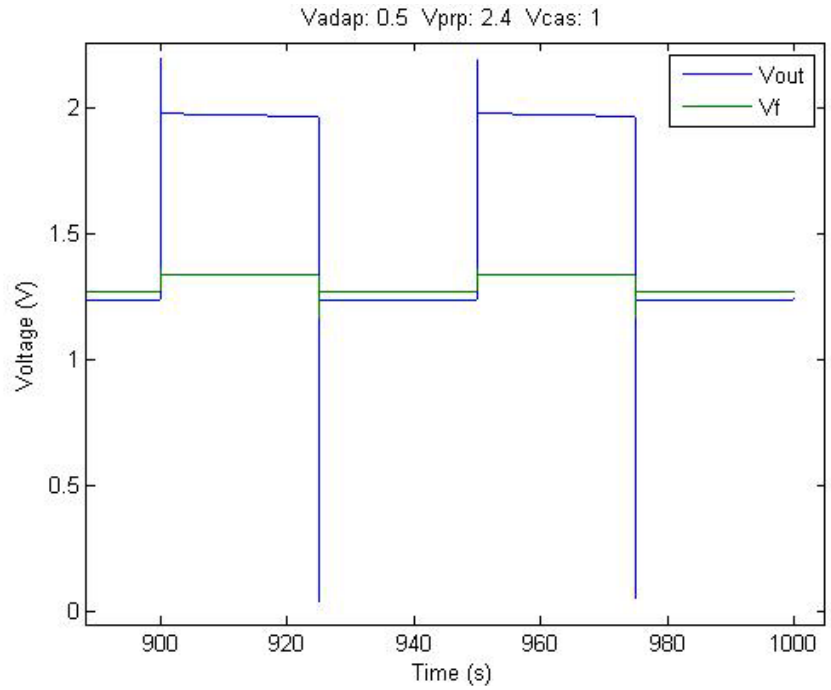

Figure 5: In this second case the adaptation bias is simply increased from the first. It is now high enough for adaptation to be seen during the "ON" section (when the LED is on), but during the low section the output and feedback voltages are too similar for significant adaptation current to flow.

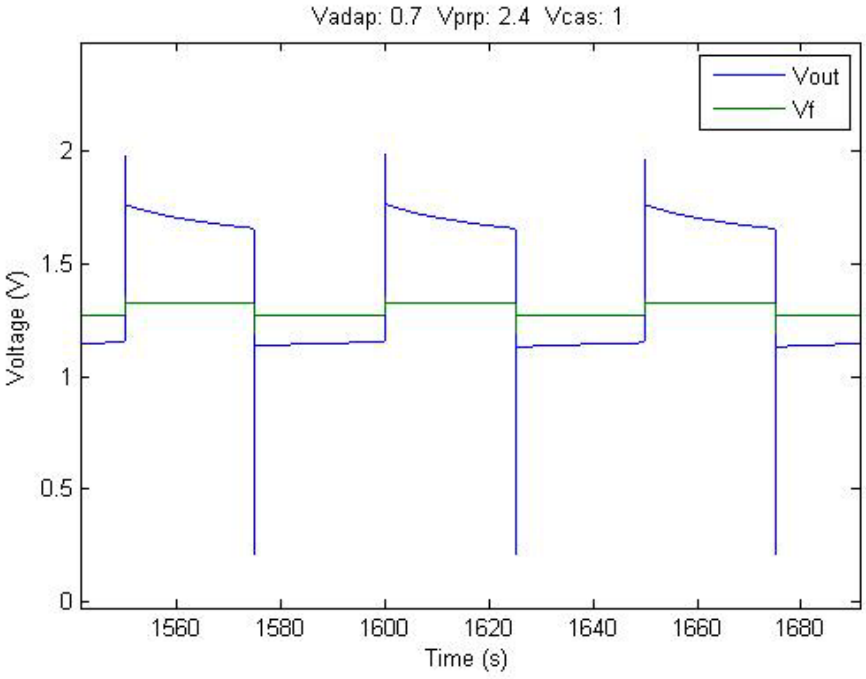

Figure 6: In the third case the adaptation has been increased even more, we now see adaptation during both the on and off sections, but the on section still has much more adaptation than the off. The off section adaptation is likely overestimated in simulation because the transistor is no longer sub-threshold, but is still being modelled using the subthreshold equations. Do you know the KP parameter (slope) for the linear region in this process? With that parameter I could better simulate the adaptation in these cases.

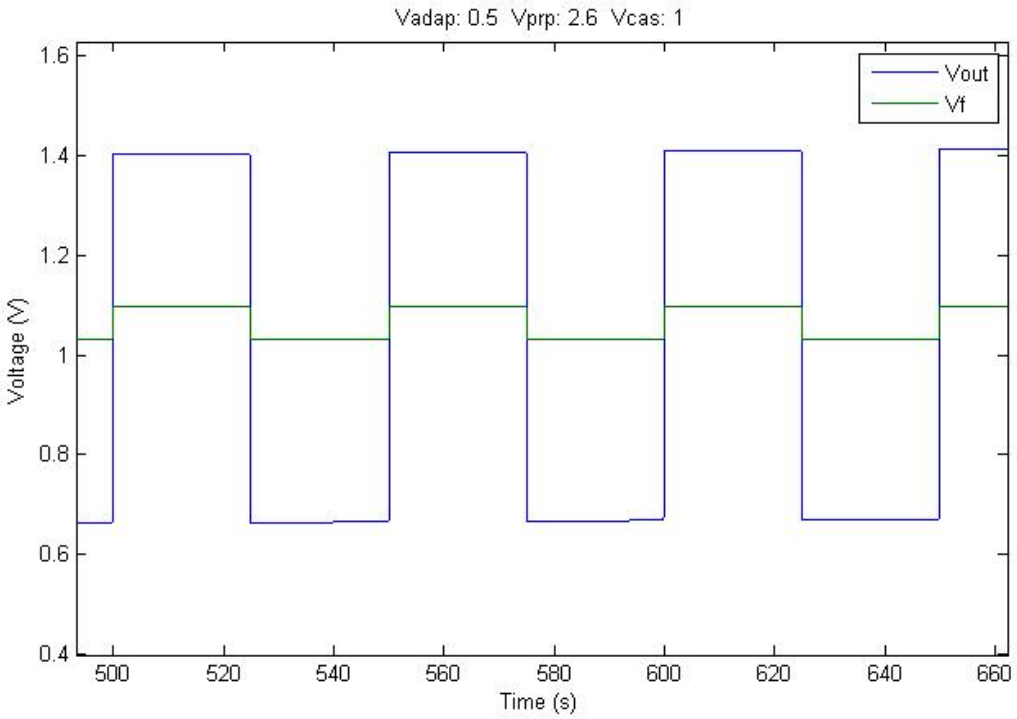

Figure 7: In the fourth case the bias current through the amplifying branch is decreased. This slows the circuit down removing the overcompensation seen in the previous two cases. Once again, the negative side adaptation is likely over-estimated.

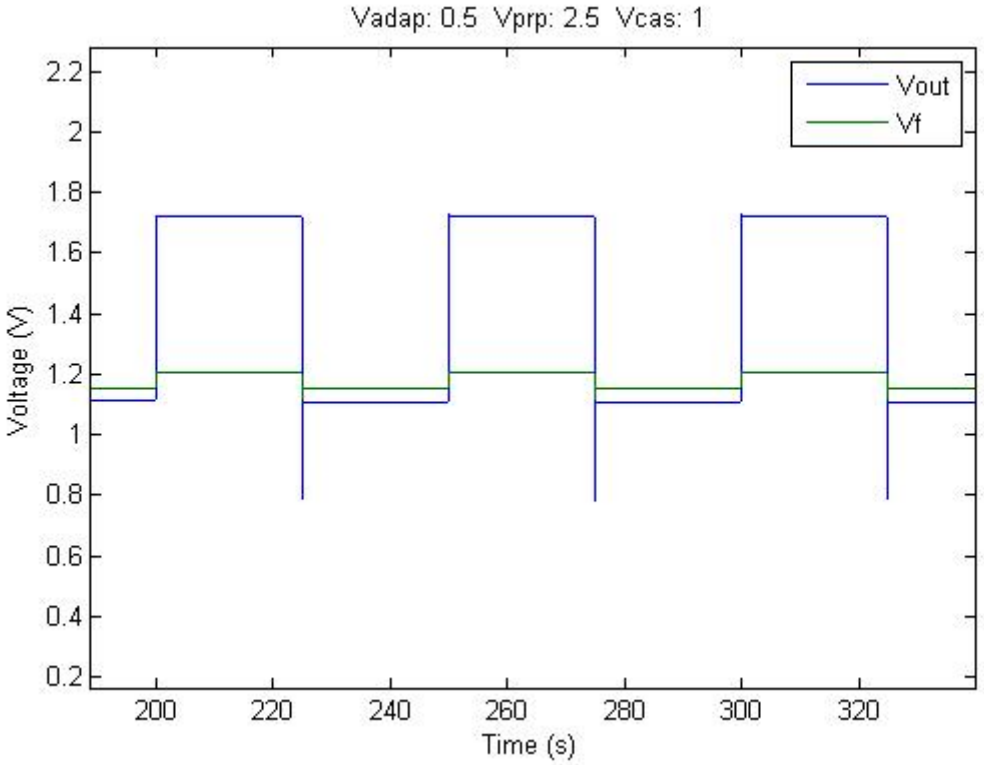

Figure 8: In the fifth case the amplifying branch bias current is such that a "spike" is only seen on the negative edge. The slope of adaptation in the off section is likely overestimated as discussed above.

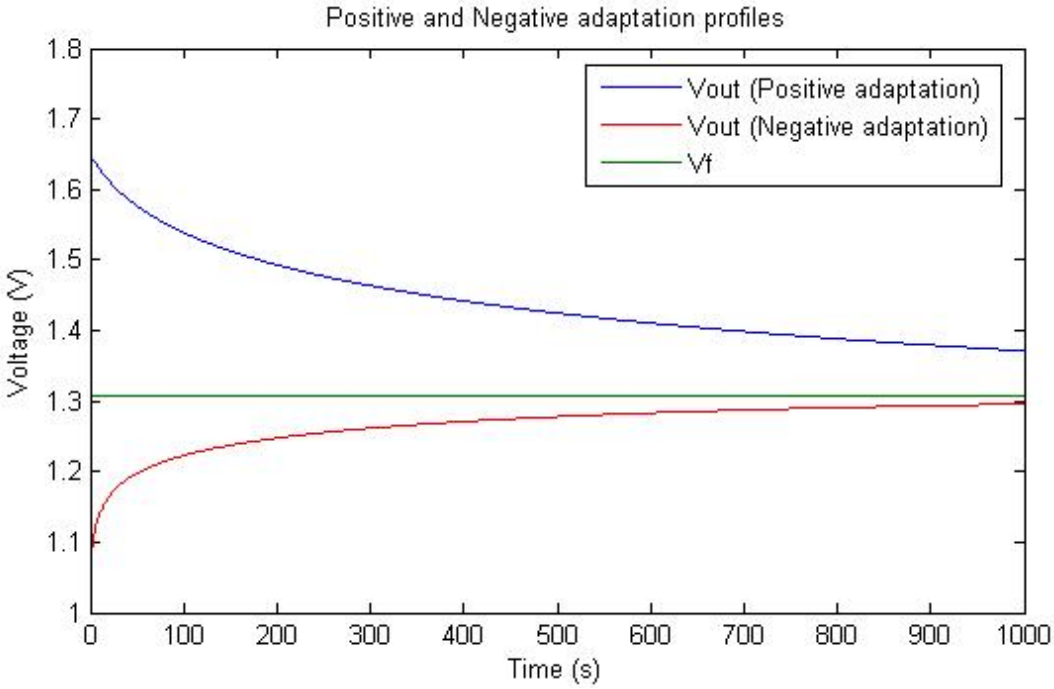

Figure 9: Plot showing adaptation profiles for the same adaptation bias when output starts from an offset of +0.3V(blue) and -0.3V(red) from steady state. This clearly shows the asymmetrical adaptation.

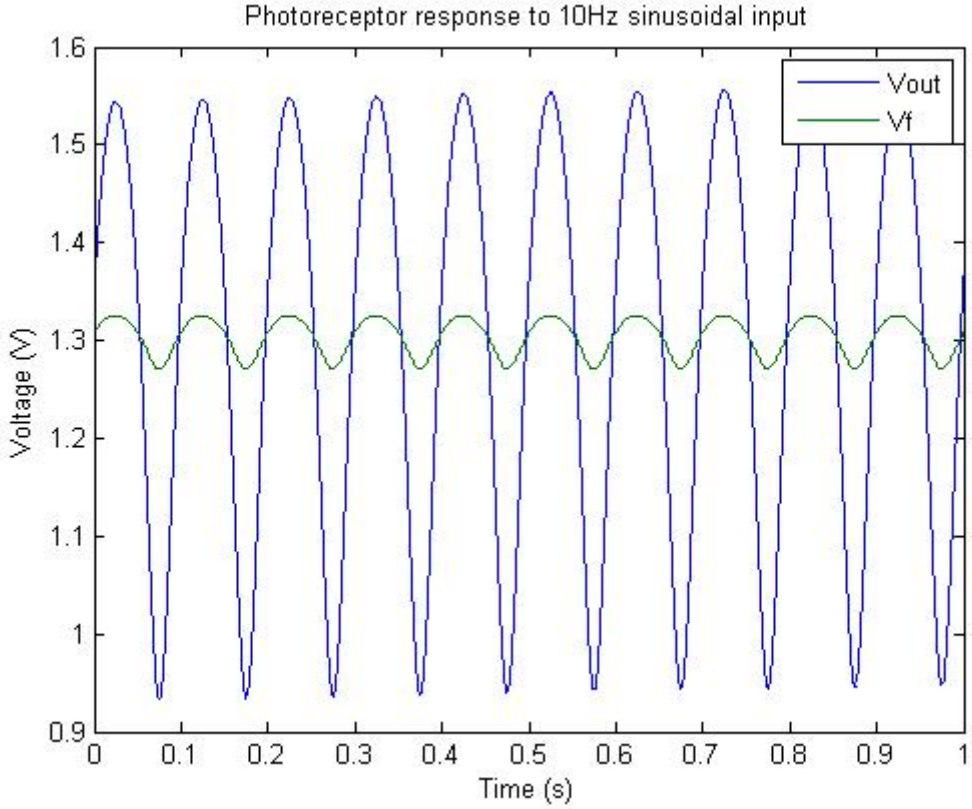

Figure 10: Photoreceptor response to a 10Hz sinusoidal input. The amplitude of the output remains roughly constant for frequencies between 10 and 100 Hz.

DISCUSSION

The photoreceptor block acts to amplify the input photocurrent and adapt to new lighting conditions. The adaptation is much faster when adjusting to darker lighting than brighter lighting. In some cases the adaptation is so strong that immediately after a decrease in lighting the circuit over-adapts resulting in a higher output than before. This has only been seen for sharp changes in lighting (a square wave), which will not be present in the images we are using for simulation.

The simulations show that decreasing the bias current can actually result in a larger signal for high frequencies because it impairs the adaptation.

After a number of simulations of the photoreceptor without high frequency inputs it was observed that the output closely reflects a scaled version of the input current. As a result the photoreceptor is modelled as a simple gain in the full chip simulations where slowly changing images are expected.

This circuit block is used to compute the derivative of the photoreceptor signal. It takes the form of a high pass active filter using an amplifier.

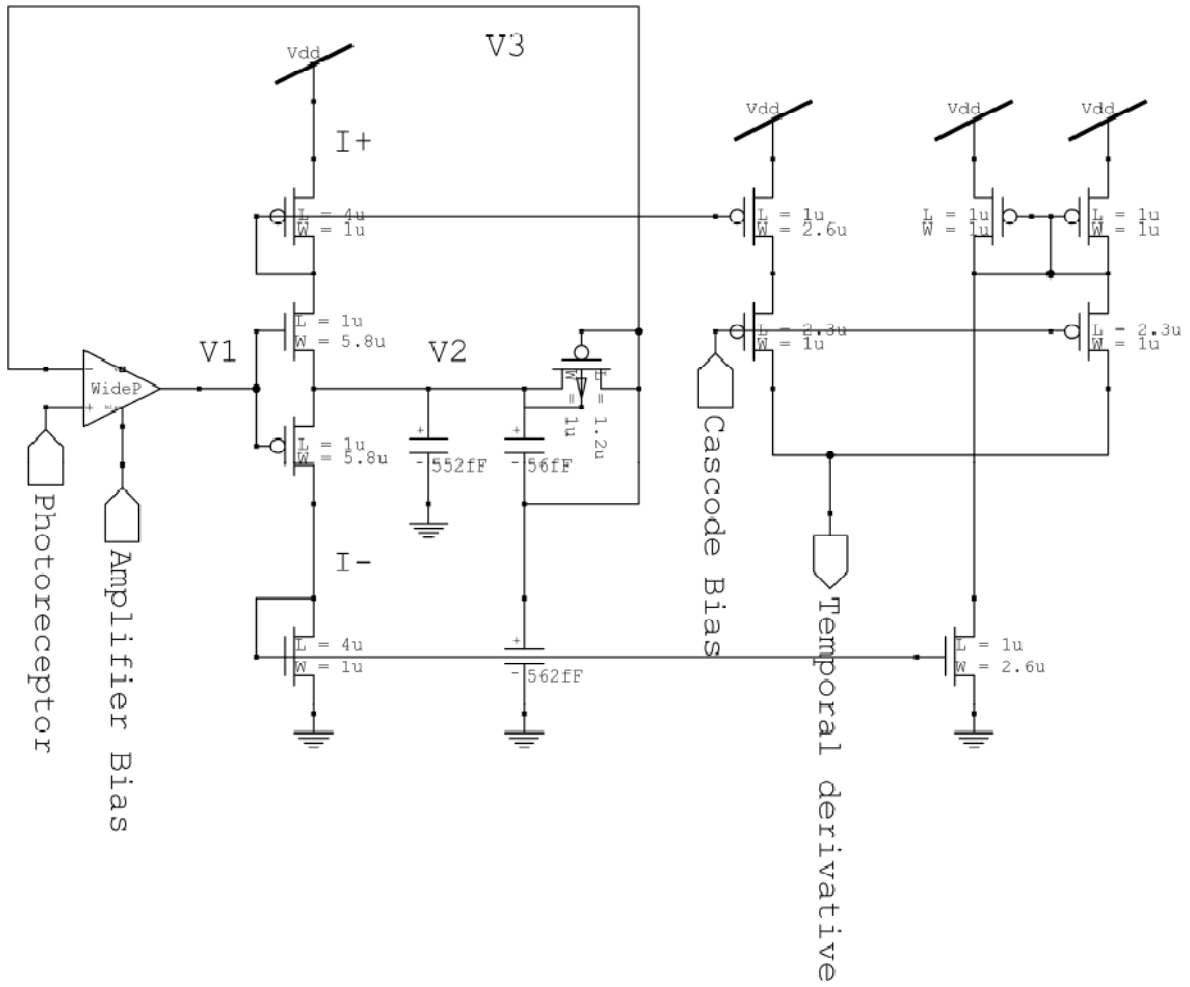

Figure 11: The circuit used to compute and rectify the derivative of the photoreceptor signal

The circuit acts as an active high pass filter to approximate the derivative of the input voltage. Due to the configuration of the inner two transistors at the output of the amplifier, only one will conduct at a time. The outer two transistors simply act to mirror the current. The five transistors on the right rectify and combine the current output using current mirrors. The resulting output is a rectified current representing the derivative of the input voltage.

SIMULATION

In order to simulate the circuit we first need to introduce the transconductance amplifier from the circuit.

AMPLIFIER

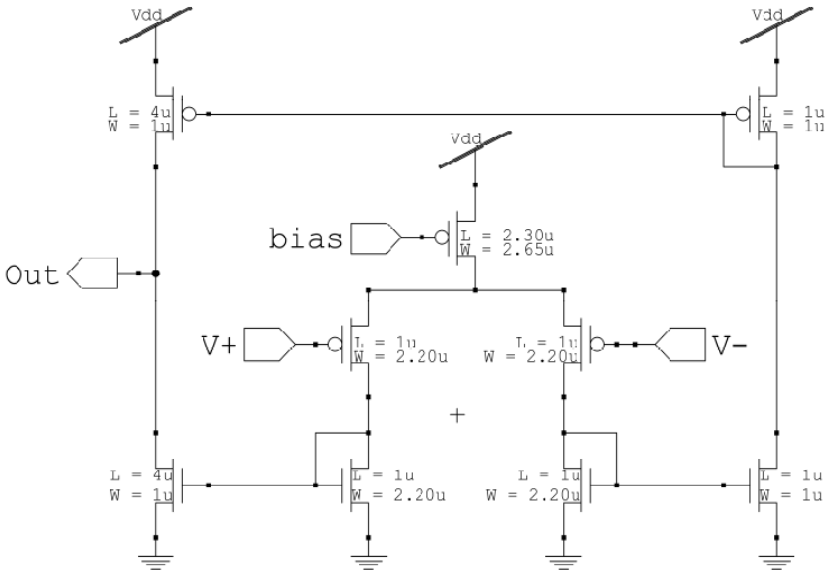

Figure 12: Wide swing PMOS differential pair amplifier

The current set by V_{bias} is split between the two inner branches depending on the relative values of V+ and V-. The currents in the two inner branches are then mirrored and subtracted to obtain the output current.

The output of the amplifier can be given by the equation

$$I_{bias} = I_{0P}\left(\frac{2.65}{2.3}\right)\exp\left(\frac{\kappa_P(3.3 - V_{Bias})}{U_t}\right)$$

$$I_{out} = \frac{I_{bias}}{(2.2)(4)}\left(\frac{1 - e^{\frac{\kappa_P(V_+ - V_-)}{U_t}}}{1 + e^{\frac{\kappa_P(V_+ - V_-)}{U_t}}}\right)$$

CIRCUIT CURRENTS

The current mirrored through the top transistor is given by

$$I_+ = \left(\frac{W}{L}\right) I_{0N} e^{\frac{\kappa_N(V_1)}{U_t}} \left(e^{-\frac{V_2}{U_t}} - e^{-\frac{3.3}{U_t}}\right)$$

The current mirrored through the bottom transistor is given by

$$I_- = \left(\frac{W}{L}\right) I_{0P} e^{\frac{\kappa_P(3.3-V_1)}{U_t}} \left(e^{\frac{V_2-3.3}{U_t}} - e^{\frac{(-3.3)}{U_t}}\right)$$

The current through the Tobi element is very roughly approximated by a transistor with its bulk tied to 3.3V and a fixed gate voltage. This provides a current which is exponentially dependant on the drain and source voltages when operating in either direction

$$I_{tobi} = \left(\frac{W}{L}\right) I_{0P} e^{\frac{\kappa_P(3.3-V_{gate})}{U_t}} \left(e^{\frac{V_2-3.3}{U_t}} - e^{\frac{V_3-3.3}{U_t}}\right)$$

The output current is simply given by

$$I_{out} = I_+ - I_-$$

DIFFERENTIAL EQUATIONS

The node voltages can then be described by

$$\frac{dV_1}{dt} = \frac{I_{amp}}{2C_{gate}}$$

$$\frac{dV_2}{dt} = \frac{(I_+ - I_-)}{552fF} - \frac{I_{tobi}}{600fF}$$

$$\frac{dV_3}{dt} = \frac{dV_2}{dt}\left(\frac{56}{56+562}\right) + \frac{I_{tobi}}{600fF}$$

RESULTS

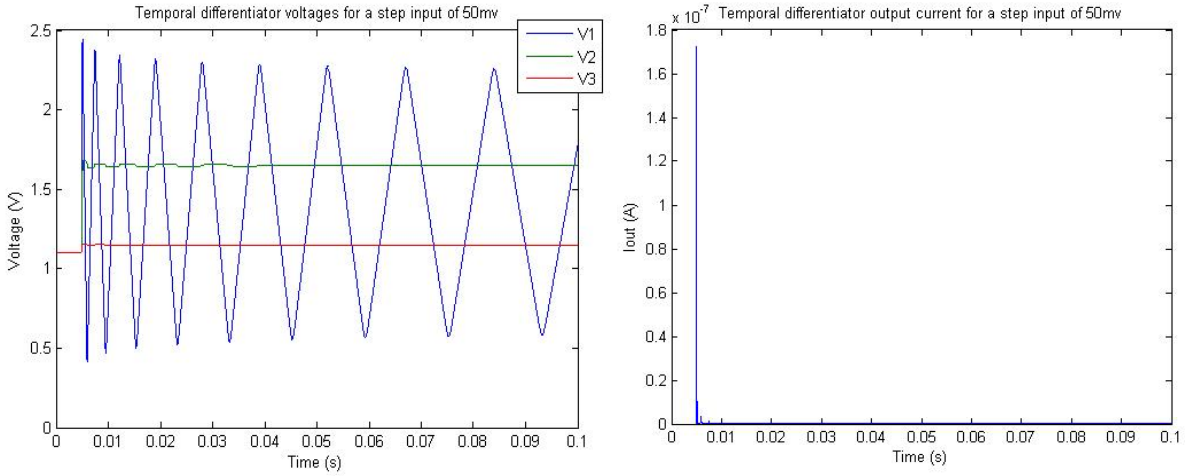

Figure 13: Temporal differentiator intermediate node voltages (left) and output current (right) after a step input of 50mV. The simulations show an oscillation of the node V1, but nevertheless the circuit appears to produce a feasible output current

DISCUSSION

The NMOS and PMOS in the current branch after the amplifier are of equal size, but due to the relatively low mobility of charge carriers in the PMOS, the NMOS is likely to dominate. This means that the circuit will be responsive to positive changes in V1, but negative changes will occur much more slowly.

Simulations results indicate a possible oscillation of the circuit output, which could prove a problem when driving the edge detection thresholding circuit as it could cause the threshold to be crossed multiple times for each image edge.

This is a tricky circuit to simulate and the oscillations in simulation may be due to the unrealistically high speed of the amplifier. Simulating the amplifier in more detail by including capacitances to slow down the speed at which the output changes could remove these oscillations, but the oscillations have a period above 1ms, which means it is unlikely that the amplifier is the cause.

A more likely cause is the relative strength of the NMOS and PMOS in the current branch on the output of the amplifier. They are of equal size, but the relatively low mobility of charge carriers in the PMOS will result in it being dominated by the NMOS. The circuit will therefore respond fast to positive changes in the amplifier output, but slowly to negative changes in the output. This slow response to negative changes can cause the circuit to oscillate.

A very rough approximation of the Tobi element has been used for simulation, but this is not the cause of the oscillations. Short circuiting the Tobi element or removing it completely both still result in oscillations.

Actual on-chip measurements of the circuit output are expected to be available soon and may help in determining what needs to be changed in simulation.

SLOW PULSE

The slow pulse block is an implementation of the Differential Pair Integrator described in [2]. The Fast Pulse output activates a current source which charges a capacitor through a differential pair after which the capacitor charge is leaked off through a discharge transistor. The charging and discharging profiles the capacitor voltage are obtained by integrating differential equations. The capacitor voltage drives the gate of an NMOS creating a current output.

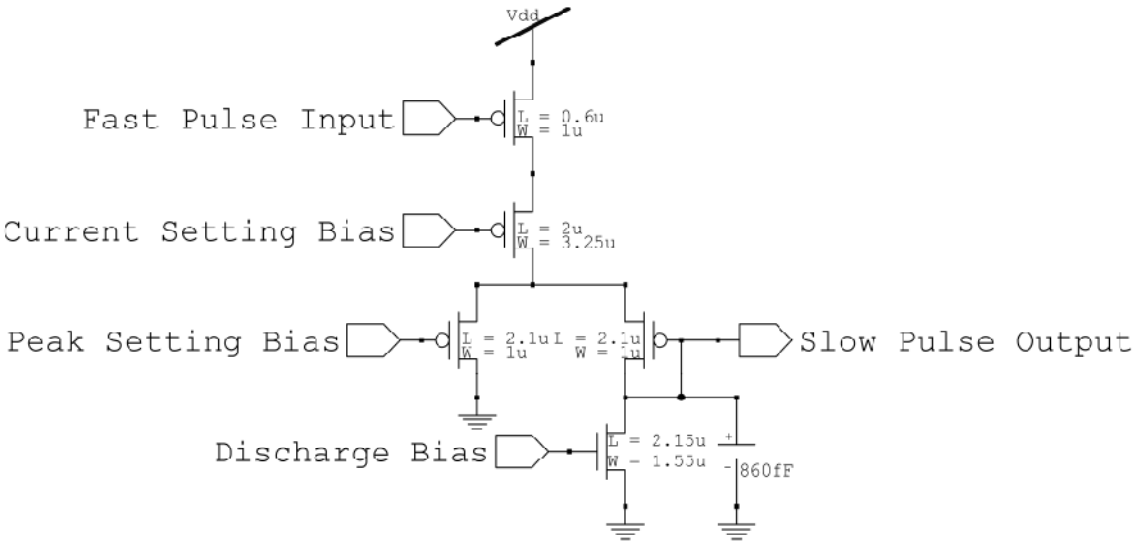

Figure 14: The Differential Pair Integrator (DPI) circuit used to generate a Slow Pulse

The Current Setting Bias acts to set the magnitude of input current to the circuit. The Fast Pulse Input acts to digitally control when the input current is preset and when it is zero. When the input current flows, it is split in the differential pair and charges the capacitor until the voltage on the capacitor is roughly equal to the Peak Setting Bias. When the input current ceases, the capacitor discharges linearly (assuming subthreshold operation) at a rate controlled by Discharge Bias.

SIMULATION

CIRCUIT CURRENTS

The current through the top transistors is given by

$$I_{bias} = \begin{cases} I_{0P}\left(\frac{W}{L}\right)_{Current\ Setting} e^{\kappa_P \frac{3.3 - V_{Current\ Setting}}{Ut}}, & Fast\ Pulse\ Input < 3 \\ 0, & Fast\ Pulse\ Input \geq 3 \end{cases}$$

The current through the right PMOS is given by

$$I_d = \frac{I_{bias}}{1 + e^{\kappa_P \frac{(V_{out} - V_{peak\ Setting})}{U_t}}}$$

The current through the discharge transistor is given by

$$I_{discharge} = I_{0N}\left(\frac{W}{L}\right)_{discharge} e^{\kappa_N \frac{V_{discharge}}{Ut}}\left(1 - e^{\left(\frac{-V_{out}}{U_t}\right)}\right)$$

DIFFERENTIAL EQUATION

The output voltage differential equation is then given by

$$dV_{out} = \frac{\left(I_d - I_{discharge}\right)}{C_{out}}$$

DETERMINING APPROPRIATE BIASES

In the “setparam” file the biases required to obtain a specific peak voltage, charge and discharge time are approximated using these equations

$$V_{discharge} = \ln\left(\frac{C_{out}\frac{V_{peak}}{t_{discharge}}}{I_{0N}\left(\frac{W}{L}\right)_{discharge}}\right)\frac{U_t}{\kappa_N}$$

$$V_{Current\ Setting} = 3.3 - \ln\left(\frac{C_{out}\frac{V_{peak}}{t_{charge}} + C_{out}\frac{V_{peak}}{t_{discharge}}}{I_{0P}\left(\frac{W}{L}\right)_{Current\ Setting}}\right)\frac{U_t}{\kappa_P}$$

$$V_{peak\ Setting} = V_{peak} - \ln\left(\frac{I_{bias}}{I_{discharge}} - 1\right)\frac{U_t}{\kappa_P}$$

The current setting bias used in simulations is actually 0.1V lower than the calculated value. This is because the charging is exponential, but is approximated as linear in the bias setting equations.

RESULTS

The charge and discharge profiles for the DPI are shown below.

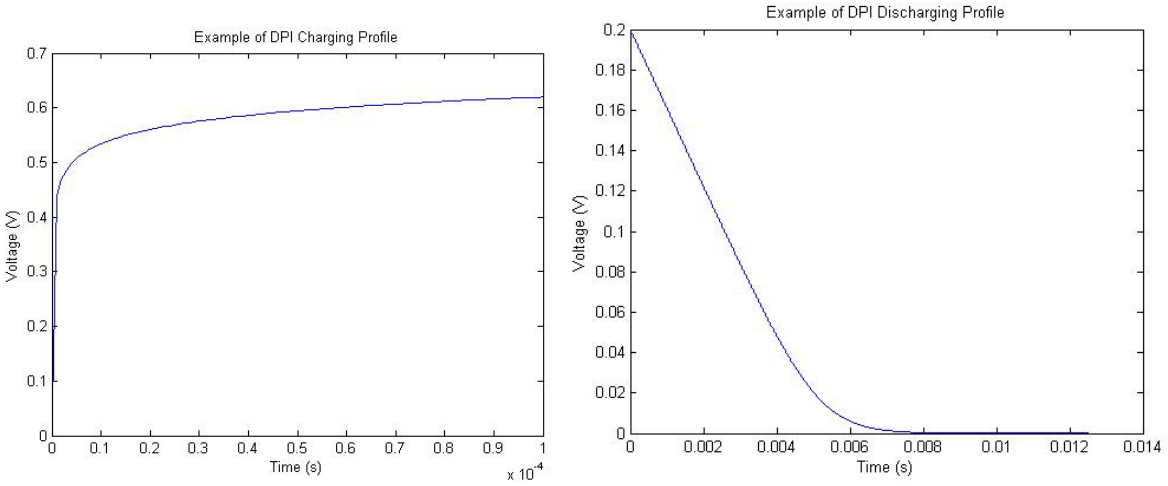

Figure 15: Slow Pulse Charging and Discharging profiles. Note that only the end of the discharging profile is shown. For voltages above 0.2V the discharge profile is approximated as a straight line with the same slope as at 0.2V.

DISCUSSION

The correct biases for the DPI are generated by the code, but these values must be checked after simulation because if extraordinarily fast charge or discharge profiles are requested the resulting biases may be such that transistors leave the subthreshold region of operation. If this happens the simulations will no longer be accurate because they will be using subthreshold equations to approximate above threshold operation.

LOW LEVEL SIMULATION

Low level simulations were implemented using approximate equations as described in the following section. The simulation allowed for a number of parameters to be varied. Most importantly the angle of the array to the velocity can be changed. The angle between the array and the velocity can greatly affect the output as discussed under the high level simulation section.

Transistor mismatch of up to 20% was introduced wherever current mirrors are present to investigate the sensitivity of the design to mismatch.

PHOTORECEPTOR

As discussed in the detailed simulations, the photoreceptor was approximated as a simple gain of its input. The input to the photoreceptors was obtained from Pangu images using five steps.

1) The mean of the image was subtracted from the original image to obtain an image with zero mean

2) The image was upsampled using interpolation to allow for a finer temporal resolution in the input signals

3) The image was low pass filtered to model the non-negligible size of each photoreceptor

4) A line of image pixels is extracted corresponding to the path of the photoreceptor moving over the image

5) The pixels are rescaled to obtain a set peak to peak output value (which is a variable in the code).

In actual simulations only the necessary portions of the image are filtered to decrease simulation time. The output peak to peak value cannot be inferred from the images because the image values themselves do not represent absolute light levels.

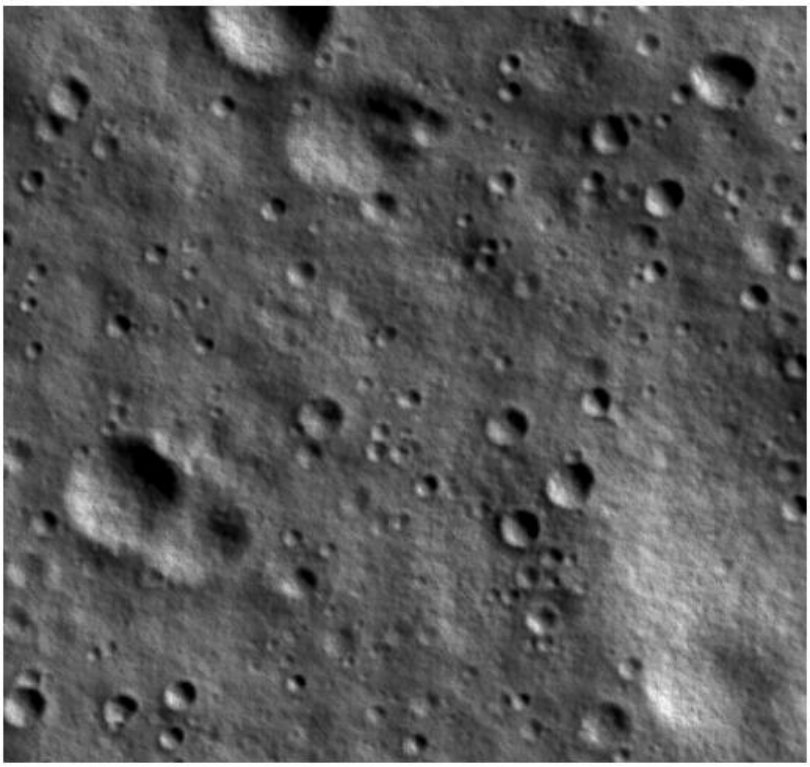

Figure 16: A Typical image generated by Pangu. This particular image was generated at a height of 2km with a Field of View (FOV) of 10 degrees.

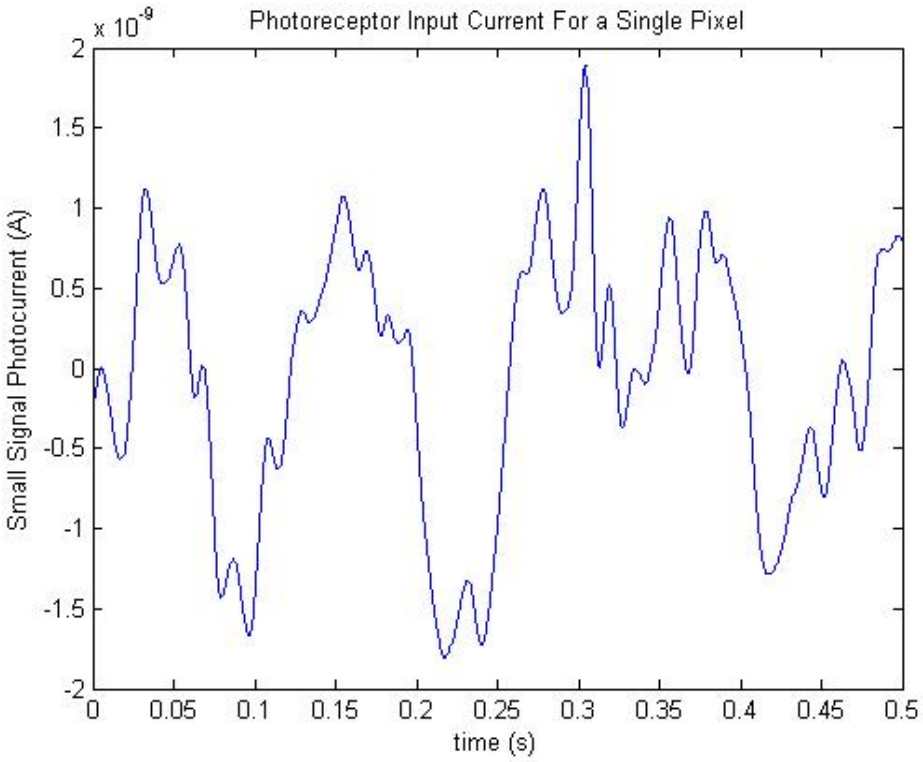

Figure 17: Photoreceptor input current for a single pixel generated from the image on the previous page.

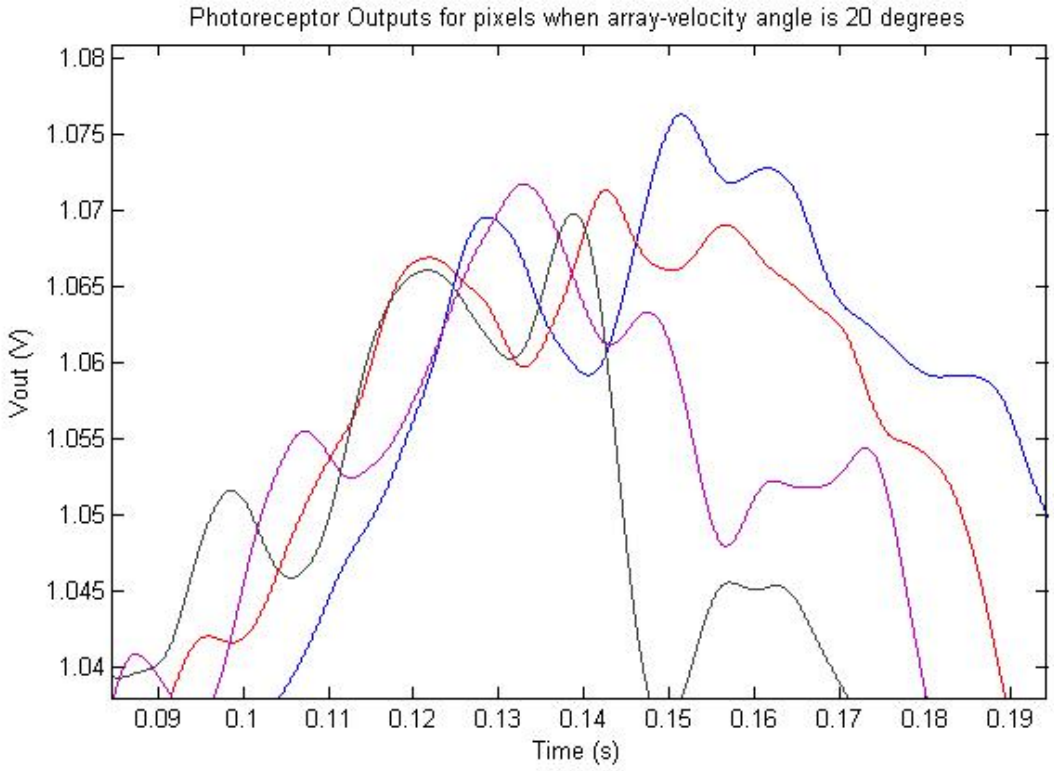

Figure 18: A subsection of the photoreceptor outputs for multiple photoreceptors with an array to velocity angle of 20 degrees. Each colour represents a different photoreceptor output. If the array to velocity angle was zero degrees the outputs would just be time delayed versions of each other.

SPATIAL DERIVATIVE

The spatial derivative block computes the difference between the outputs of the photoreceptors on the left and right of the current pixel, thus giving a measure of the spatial derivative. The circuit is known as the bump-antibump circuit and is described in [1].

CIRCUIT

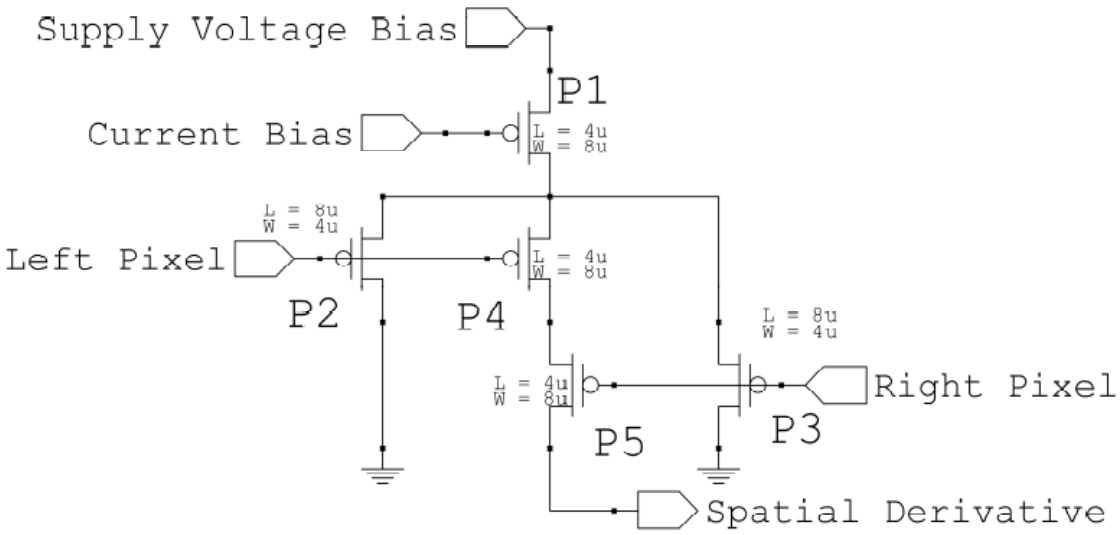

Figure 19: Circuit diagram showing the bump anti-bump circuit used to approximate the spatial derivative

SIMULATION

The current output of the circuit is given by the following equations

$$I_{bias} = I_{0P}\left(\frac{W}{L}\right)_{P1} \exp\left(\frac{\kappa_P\left(V_{Supply\ Voltage\ Bias} - V_{Current\ Bias}\right)}{U_t}\right)$$

$$I_{Spatial\ Derivative} = \frac{I_{bias}}{1 + \frac{4}{S}\left(\cosh\left(\kappa_P \frac{\left(V_{Left\ Pixel} - V_{Right\ Pixel}\right)}{2}\right)^2\right)}$$

Where S is the $\frac{W}{L}$ ratio of P4, P5 to P2, P3 (which in this case is 4)

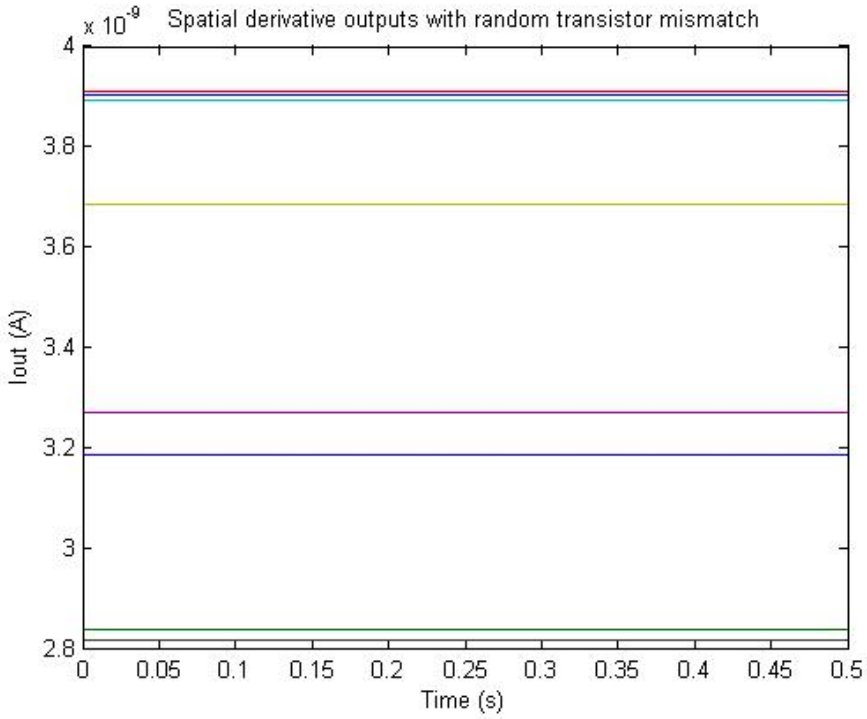

Figure 20: Spatial derivative outputs for an 8 pixel simulation with transistor mismatch. Each color represents the spatial derivative measured at a single pixel. The plot clearly shows how the spatial derivative circuit output is dominated by transistor mismatch.

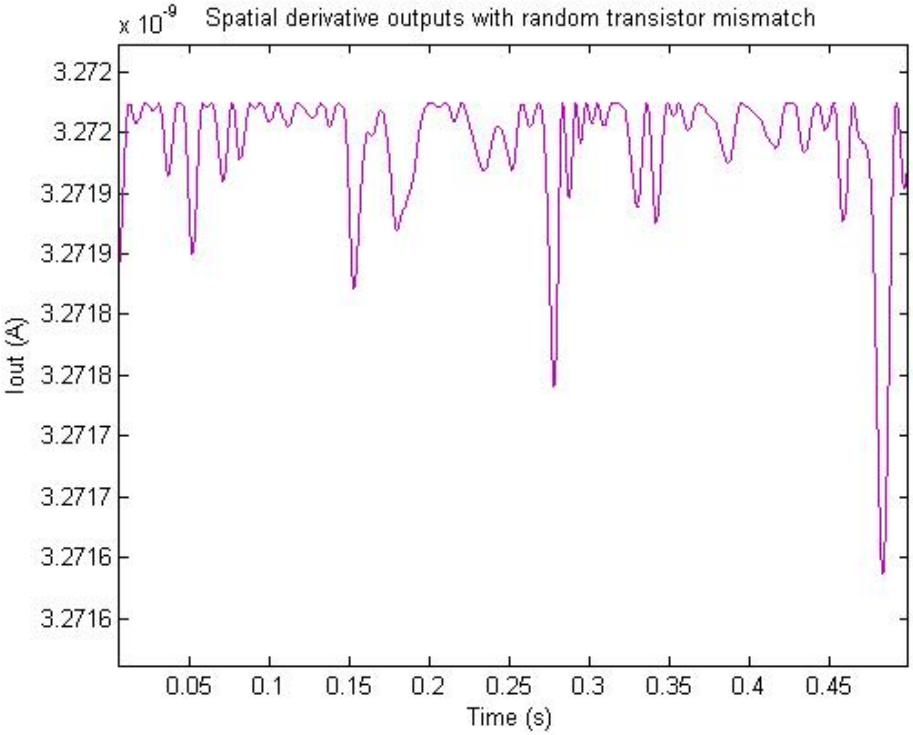

Figure 21: A closer view of a single pixel's spatial derivative output showing that a signal is present, but it is swamped by the transistor mismatch effect shown in the previous plot.

According to the above equations a 10% change in the output current would require a difference in input current of 1.3V. Such high signal differences are unlikely, especially in the planetary landing scenario. The signal is therefore likely to be very small compared to the offset. This makes the circuit very sensitive to transistor mismatch, particularly for the bias setting transistor.

The spatial difference block is therefore unlikely to provide a useful input to the winner take all if transistor mismatch is taken into account. A one percent change in signal requires 0.4V difference between the input voltages, which is very unlikely, thus even if transistor mismatch is only at 1% it is likely to dominate the output signal.

TEMPORAL DERIVATIVE

This circuit block outputs a scaled and rectified derivative of the photoreceptor output voltage [1]. The detailed simulations must still be verified against actual chip measurements.

SIMULATION

In its most simple form the circuit output can be approximated as the scaled, rectified derivative of the input. This is the equation used in the full chip simulations.

$$I_{out} = \left| C \frac{dV_{photoreceptor}}{dt} \right|$$

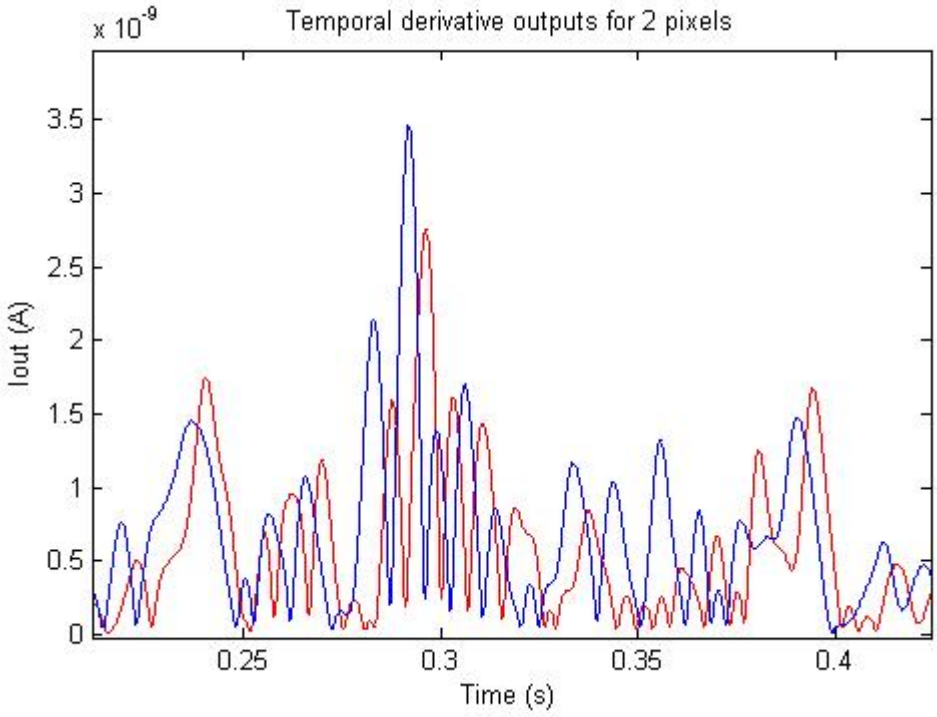

Figure 22: Temporal Derivative outputs for 2 pixels showing how the temporal derivatives vary between pixels when the array to velocity angle is non-zero (20 degrees in this case).

The edge threshold block compares the temporal derivative to a set threshold and outputs a current when the threshold is crossed. The circuit takes the form of a winner take all circuit [1].

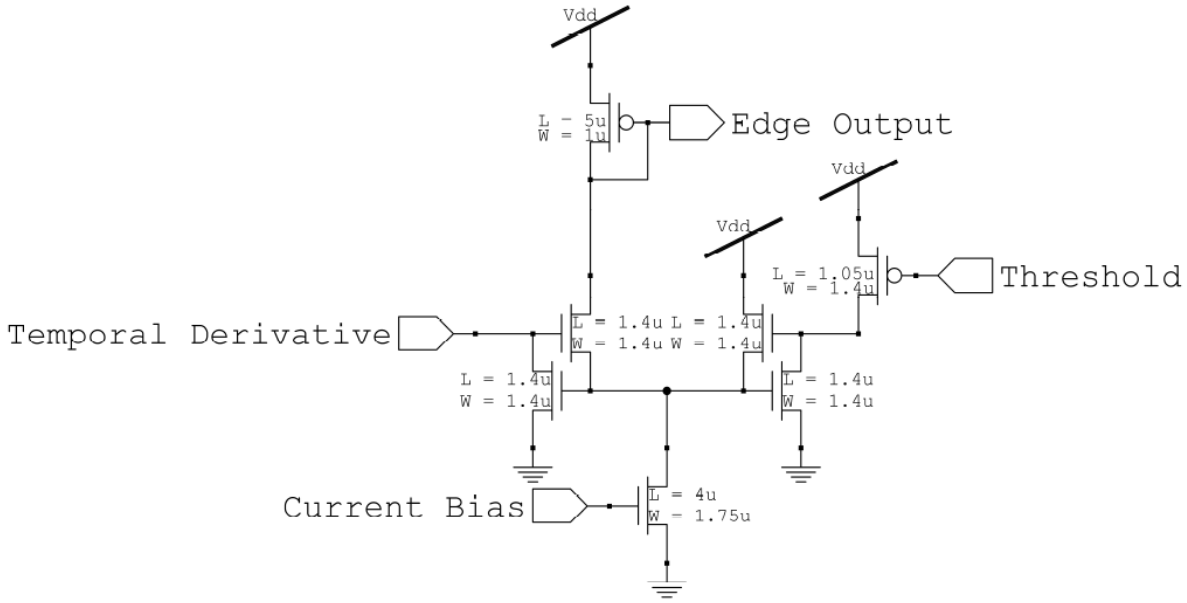

Figure 23: A two input Winner-Take-All (WTA) circuit used as a thresholding circuit

When the Temporal Derivative input current is greater than the threshold input current then an output current is drawn through the top transistor. The voltage on Edge Output can be used to mirror this current.

The equations for the winner take all circuit can be found in [1]. They are repeated here for completeness.

$$\Delta V_g = \frac{V_e\left(I_{Temporal\ derivative} - I_{Threshold}\right)}{I_{sat}}$$

$$I_{bias} = I_{0N}\left(\frac{1.75}{4}\right)\exp\left(\frac{\kappa_N(V_{Current\ Bias})}{U_t}\right)$$

$$I_{out} = \left(\left(\frac{I_{bias}}{2}\right)e^{\frac{\Delta V_g \kappa_N}{U_t}}\right)$$

Due to the manner in which the output is used, it can be approximated as a digital output with value 1 whenever the input is greater than the threshold current.

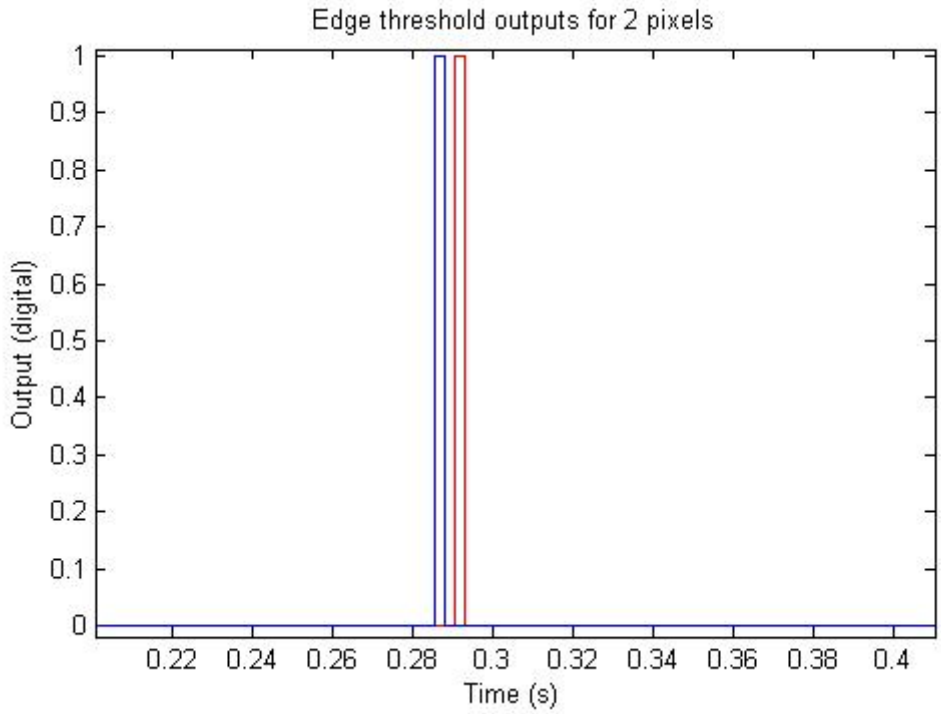

Figure 24: Digital edge outputs for two pixels

FAST PULSE

The Fast Pulse block outputs a fixed length fixed voltage square pulse when it receives a current input from the Edge Threshold block.

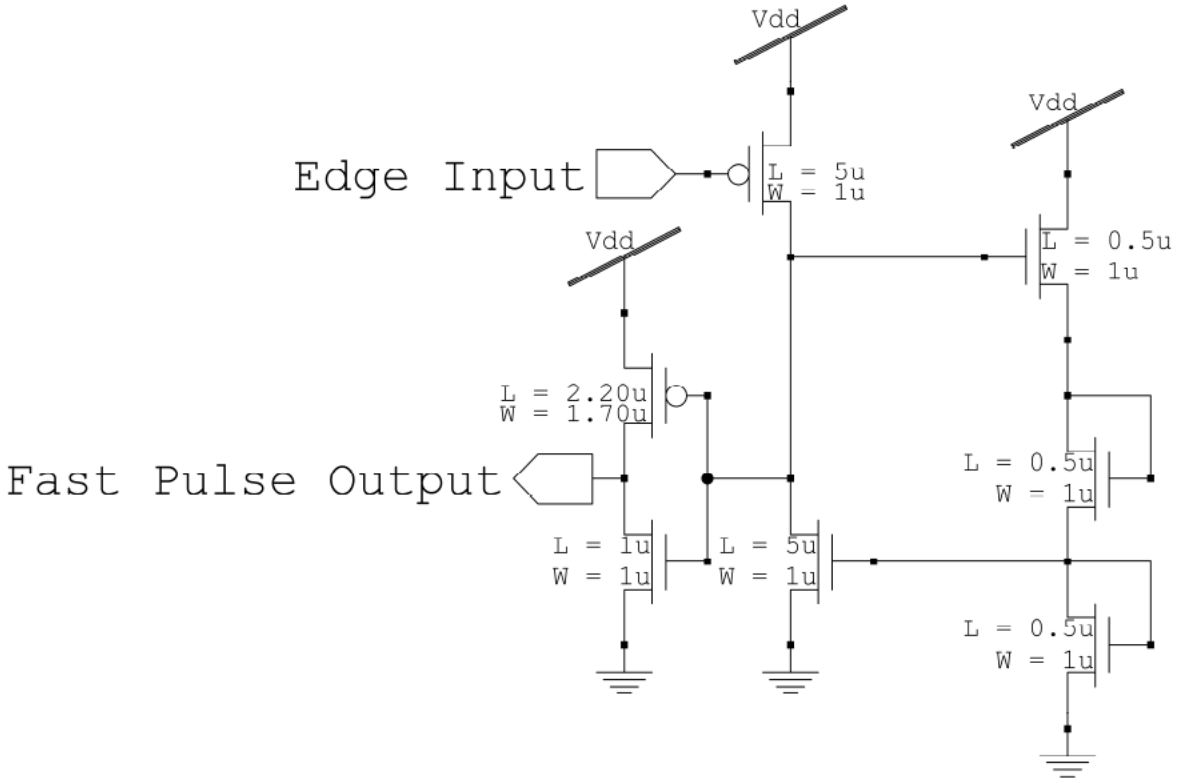

Figure 25: Circuit used to generate a fixed length digital pulse

The input voltage *'Edge Input'* is used to mirror the current from the threshold detecting block. If the current is zero, then the circuit will move towards a state where the input to the output inverter is low, thus the circuit has a high output in the resting state.

When Edge input mirrors a current this current charges the inverter input causing the inverter output to go low. Charging the inverter input node also increases the current in the right branch, which then is mirrored to the left branch via the bottom transistors where it draws current from the inverter input node, thus discharging the inverter input causing it to go high again.

When the input current to the circuit goes to zero, the intermediate node will continue to discharge and the output will remain low.

In simulation this circuit simply gives a fixed length pulse when the edge circuit block output is high. This pulse length was set to 100 microseconds in simulation, but must be measured on the chip as it cannot be controlled with a bias.

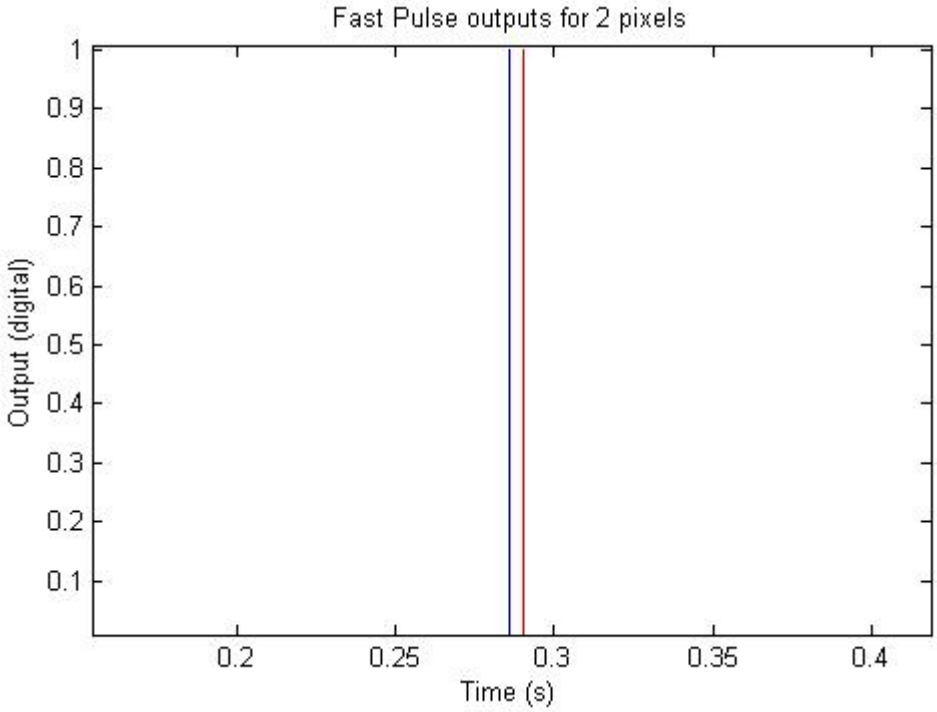

Figure 26: Fast Pulse circuit output showing the fixed length pulses generated for two pixels when they cross threshold

The Slow Pulse block consists of a DPI as discussed earlier. In the full chip simulations the DPIs (one for each pixel) are simulated only once using differential equations to obtain the charge and discharge profiles. The full chip simulations then determine the DPI output based on the input voltage from the Fast Pulse block and the pre-generated charge and discharge profiles.

The Slow Pulse block for each pixel has the same bias voltages, but transistor mismatch causes the actual current values to differ. In simulation this is modelled as a random variation of up to 20% in the transistor W/L ratio. The random number is pulled from a uniform distribution over the range [-20%, +20%].

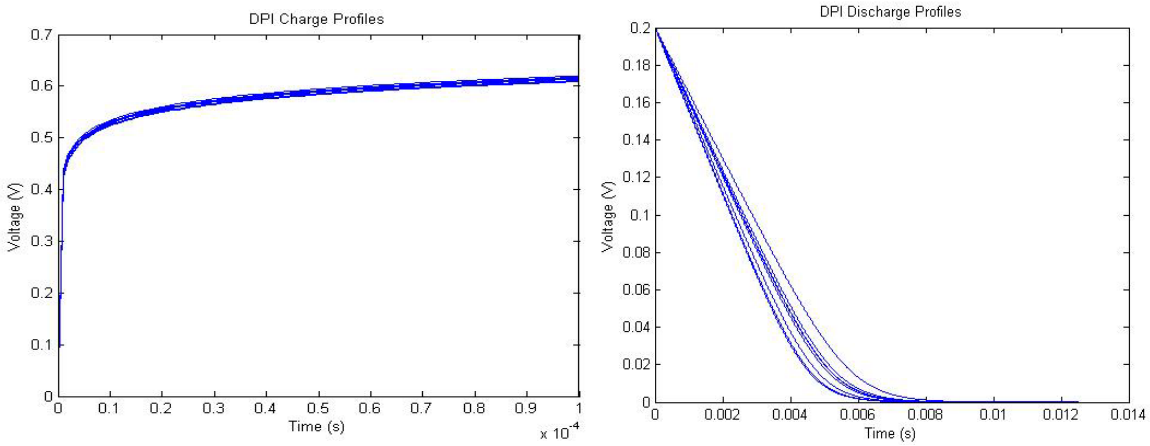

Figure 27: Slow Pulse charge (left) and discharge (right) profiles for 8 instances of the DPI (one for each pixel). The plots show how the discharge profiles vary between pixels due to noise

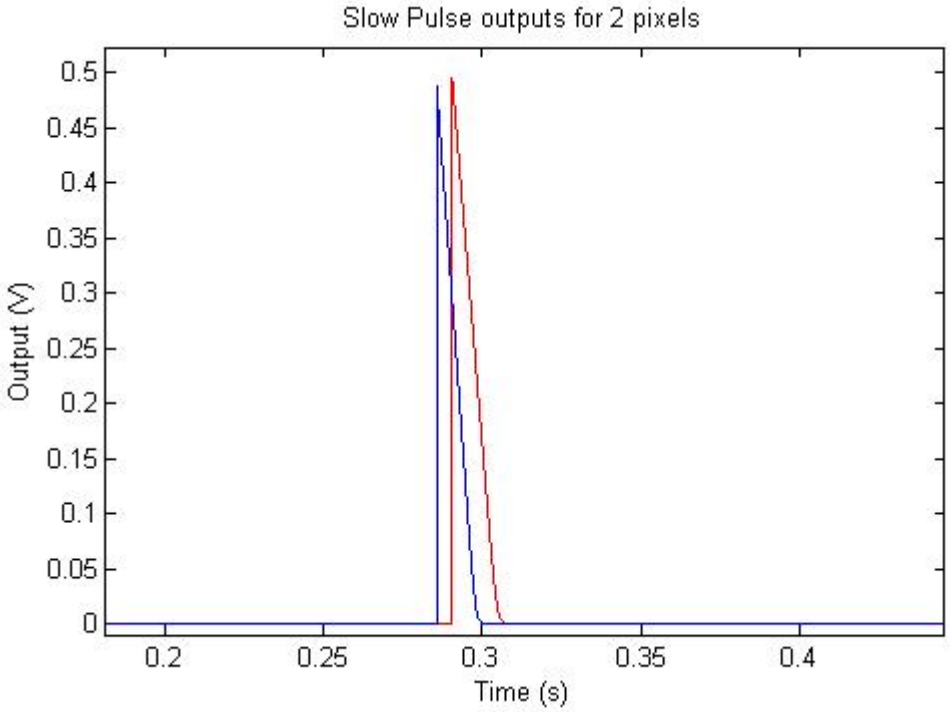

Figure 28: Slow Pulse voltage output for 2 pixels. Where they cross is where the blue signal will be sampled

SAMPLE AND HOLD

When the Fast Pulse output from a pixel to the left or right of the current pixel is asserted, the Sample and Hold block samples the output of the Slow Pulse block. This gives a measure of the time taken for an edge to cross from the current pixel to an adjacent pixel (edge velocity). For each pixel there are two Sample and Hold outputs. One sampled when the Fast Pulse output of the pixel to the left is asserted and one when the output to the right is asserted.

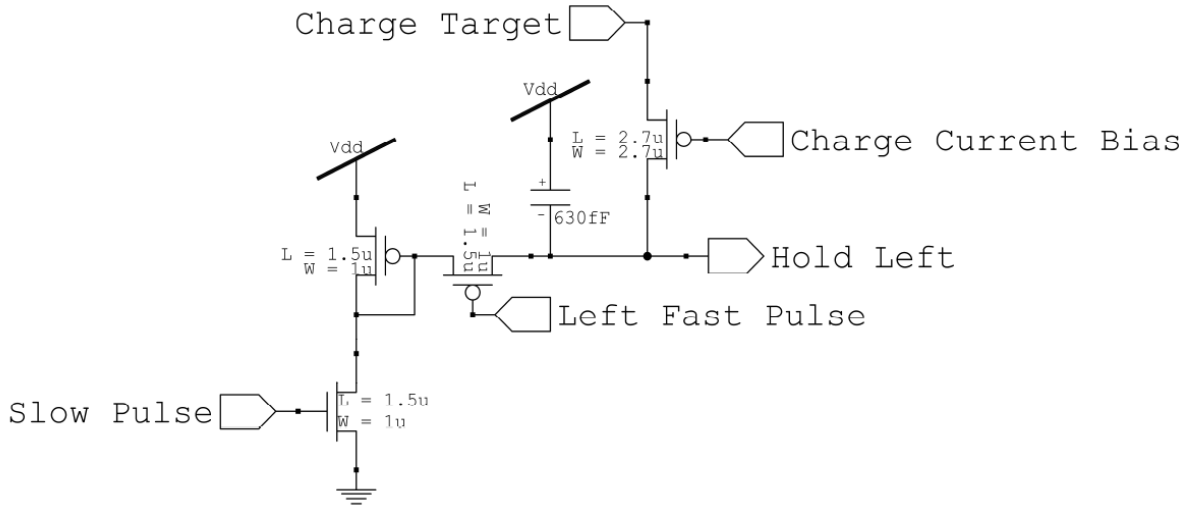

Figure 29: Sample and hold circuit

The Slow Pulse input voltage is converted to a current using the bottom transistor. This current can be mirrored using the gate voltage of the PMOS above. During a digital pulse from the Fast Pulse block of an adjacent pixel, the capacitor voltage will charge to the current mirroring voltage.

SIMULATION

In simulation the output voltage of the Slow Pulse block is converted into a current using the bottom transistor and the Sample and Hold circuit simply samples this current whenever the Fast Pulse line is active. It is assumed that simulation time is small enough for the sampled voltage to not change significantly.

The voltage to current conversion is given by the following equation :

$$I_{Slow} = \begin{cases} I_{0N}\left(\frac{W}{L}\right) e^{\kappa_N \frac{V_{Slow\ Pulse}}{U_T}}, & V_{Slow\ Pulse} < V_{thr} \\ KP_N \left(\frac{W}{L}\right)\left[(V_{Slow\ Pulse} - V_{thr})V_d - \frac{V_d^2}{2}\right], & V_{Slow\ Pulse} \geq V_{thr}\ and\ \ V_d < V_g - V_{thr} \\ \frac{KP_N}{2}\left(\frac{W}{L}\right)(V_{Slow\ Pulse} - V_{thr})^2, & V_{Slow\ Pulse} \geq V_{thr}\ and\ \ V_d > V_g - V_{thr} \end{cases}$$

DISCUSSION

In simulation it is assumed that $V_d > V_g - V_{thr}$, but this is not necessarily true. Furthermore, even if it is true, the current mirror is likely to perform poorly. For equal currents to flow through the NMOS and PMOS $(V_{SG} - V_{thr})$ of the PMOS must be about three times higher than $(V_{GS} - V_{thr})$ for the NMOS, this can cause the drain voltage of the NMOS to drop until it operates in the linear region, which would produce undesirable results. Because of this the peak output voltage of the Slow Pulse Block is typically set to below the NMOS threshold voltage to prevent this scenario.

Further care must be taken because even if the NMOS is below threshold, it could source enough current to pull the PMOS above threshold. When attaching the gate of an input PMOS (in the direction circuit on the next page) to the "Hold Left" output, this would result in the mirrored current being dependant on the drain voltage of PMOS in the direction circuit, which will provide undesirable results.

To lessen this effect the PMOS used to mirror the current must be made larger to ensure it requires a smaller V_{SG} to source the same current.

The actual values in simulation are not necessarily correct because currently the KP_N value and threshold values are approximate and not specific to the process.

DIRECTION

The direction block is a three input winner take all circuit. One input is a pre-set threshold and the other two are the outputs from the Sample and Hold block. If either of the Sample and Hold outputs are above the threshold then the greater of the two will be passed to the final winner take all as the velocity measurement. The circuit also outputs which sample was greater, indicating the direction of motion of the edge. If the threshold is the greatest input then none of the outputs are asserted.

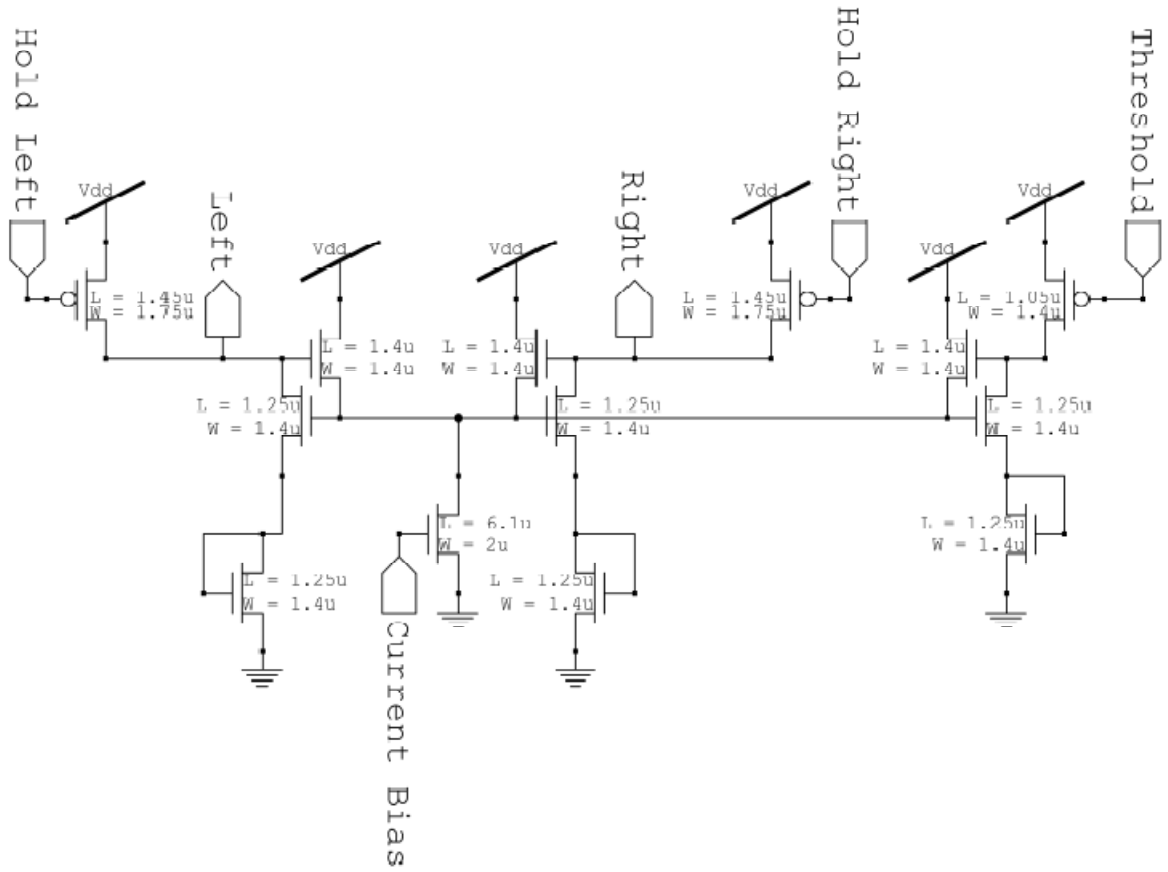

Figure 30: Winner take all circuit for determining direction

SIMULATION

In Simulation the circuit was approximated using the following equations:

$$I_{thresh} = I_{0P}\left(\frac{W}{L}\right)_{thresh} e^{\kappa_P \frac{3.3 - V_{thresh}}{U_t}}$$

V_{left} and V_{right} are effectively digital signals. They will not reach 3.3V or 0V, but can be treated as if they do because they are later passed through a level restoring inverter

$$V_{left} = \begin{cases} 3.3, & I_{left} > I_{right} \text{ and } I_{left} > I_{thresh} \\ 0, & otherwise \end{cases}$$

$$V_{right} = \begin{cases} 3.3, & I_{right} > I_{left} \text{ and } I_{right} > I_{thresh} \\ 0, & otherwise \end{cases}$$

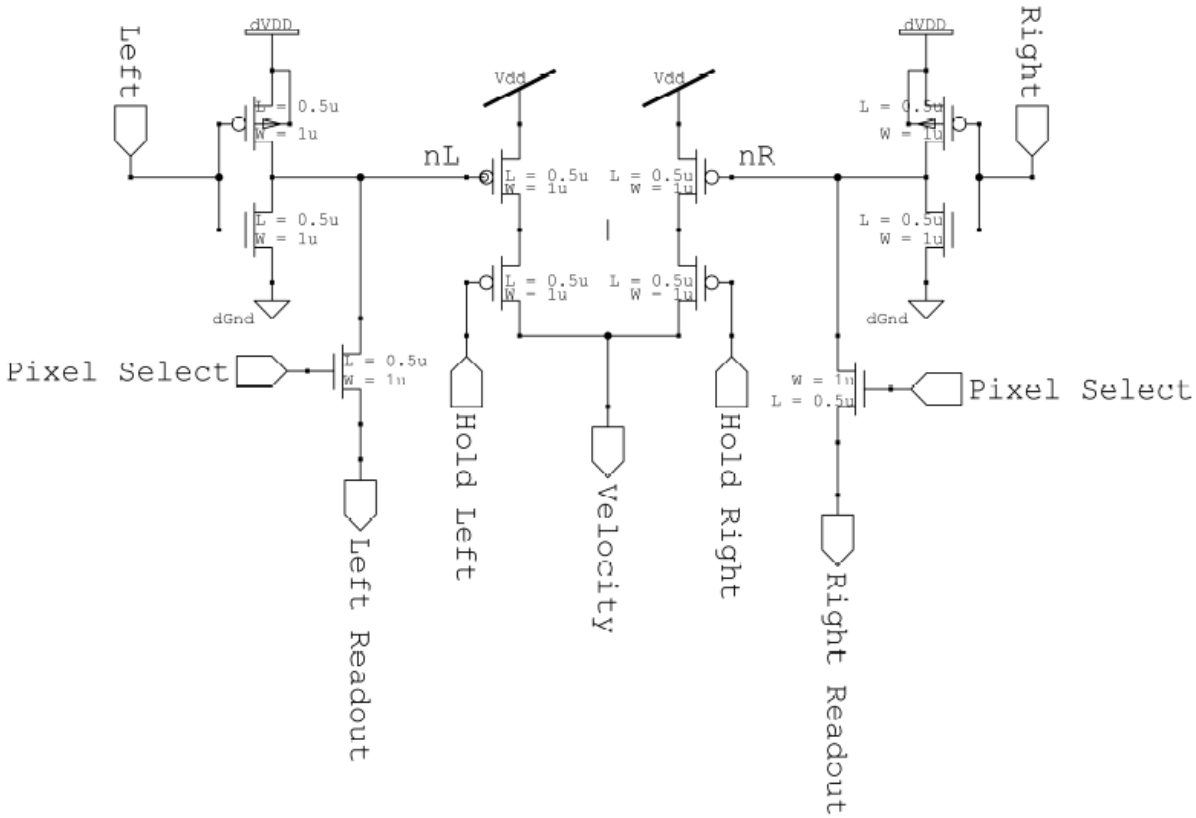

Figure 31: Circuit to pass either left or right velocity reading

In the above circuit the “Left” and “Right” inputs come from the winner take all circuit which ensures that at most one of them will be asserted high at a time. When one is asserted high it turns on the corresponding PMOS which acts as a switch to pass through the current which is determined by “Hold Left” or “Hold Right”. The sizing of the transistors results in the velocity current being scaled down by a factor of 3.

$$I_{Velocity} = \begin{cases} \frac{1}{3} I_{Slow(left)}, & if\ V_{Left} = 1 \\ \frac{1}{3} I_{Slow(right)}, & if\ V_{right} = 1 \\ 0, & otherwise \end{cases}$$

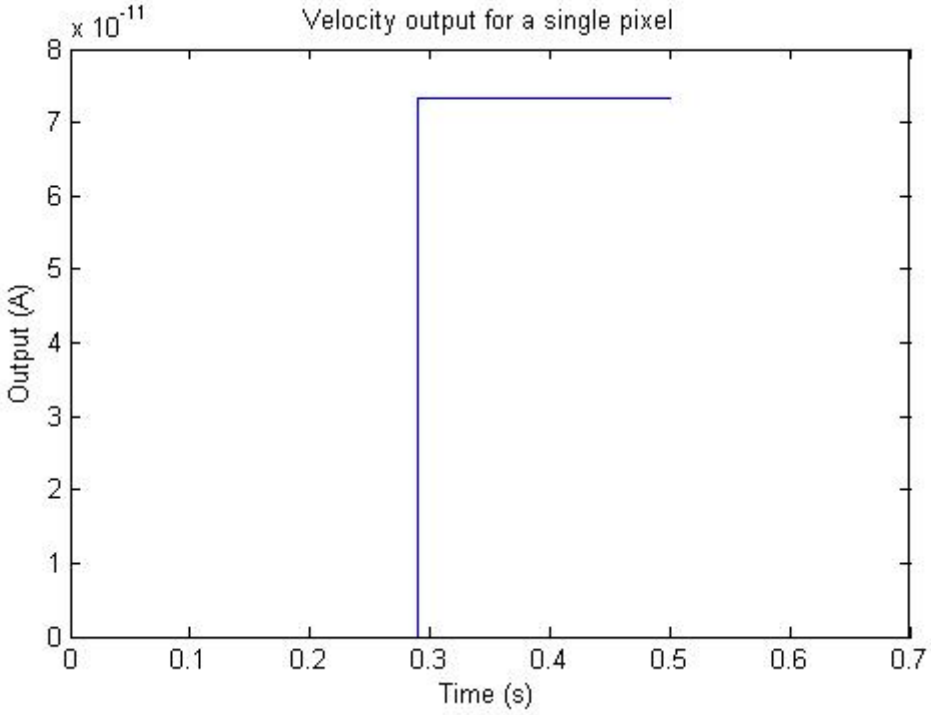

Figure 32: Velocity output current from the direction block for a single pixel

DISCUSSION

The winner take all can be modelled in more detail using similar equations to the two-input winner take all used for thresholding, but using the complete equations is unlikely to significantly alter the results. The current mirror used in the above circuit could possibly be improved by placing the PMOS switches at the drains of the mirroring transistors rather than at the source.

CURRENT NORMALISER

A discussion on the current normaliser circuit can be found in [1]. There are three current normaliser circuits, each with an input from every pixel. One has the spatial derivatives as inputs. Another has the temporal derivatives as inputs and the third has the outputs from the Direction block as inputs. As the name suggests, the outputs of each current normaliser circuit is a scaled value of its inputs such that the sum of the outputs is equal to a predetermined value (set by an external bias). Fig 2 shows how the current normalisers are used in generating the Saliency Map.

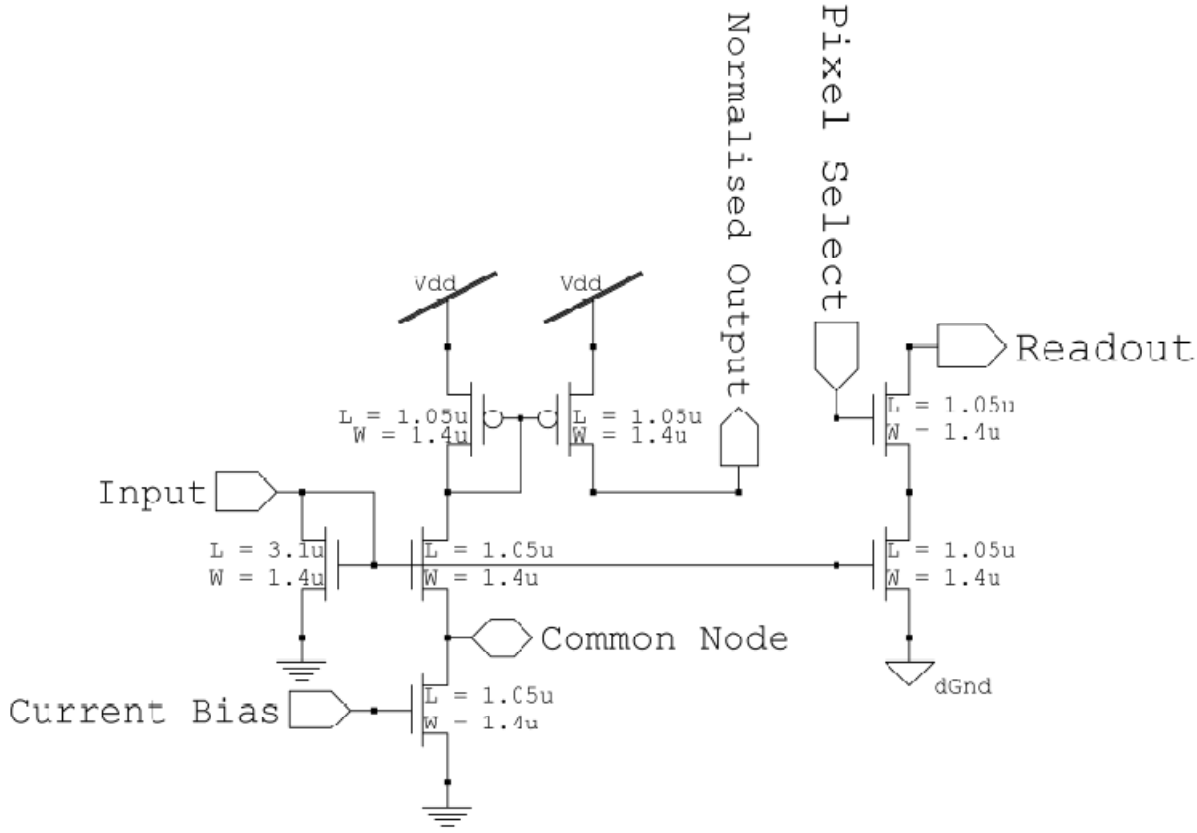

Figure 33: Current normaliser circuit

SIMULATION

In simulation the current normaliser is modelled using the following equations

$$I_{out(i)} = \frac{I_{out(i)} \sum_{i=1}^{numPixels} I_{bias}}{\sum_{i=1}^{numPixels} I_{out(i)}}$$

SALIENCY MAP

The saliency map for each pixel is the sum of that pixel's normalised spatial derivative, temporal derivative, velocity measurement minus the output of the inhibition of return circuit (discussed below). The relative weighting of the inputs can be controlled by changing the biases of the current normalisers. An advantage of current mode design is that addition and subtraction can be easily implemented by connecting the current outputs to a common node.

WINNER TAKE ALL

The Saliency Map output is fed to a winner take all circuit with lateral excitation and hysteresis [4]. The output of the winner take all determines which pixel's readings to consider. Hysteresis ensures smooth transitions between pixels while the lateral excitation promotes transition to an adjacent pixel as an edge propagates across the photoreceptors.

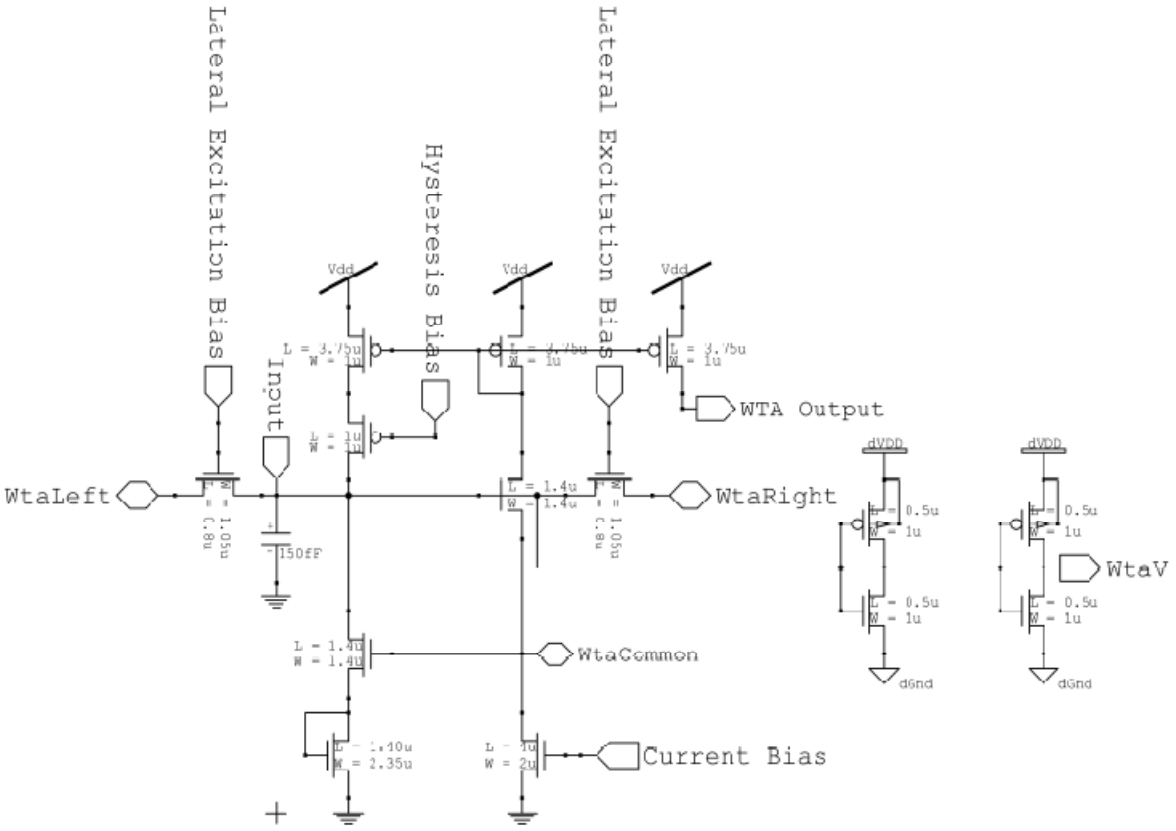

Figure 34: Winner take all with lateral excitation and hysteresis

SIMULATION

In simulation the lateral excitation and hysteresis were not implemented. The simulated circuit simply outputs a current $(numPixels)(I_{bias})$ at the pixel output corresponding to the highest pixel input.

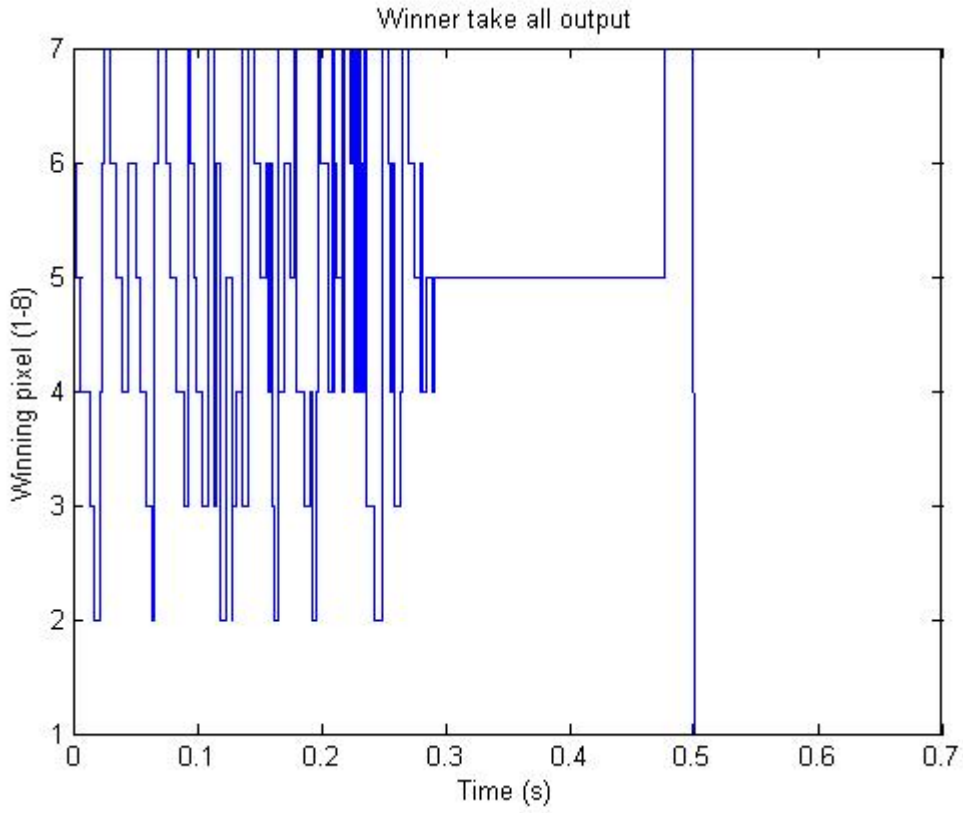

Figure 35: Winner take all output for an 8 pixel simulation. Initially the winner take all jumps between pixels until just before 0.3s when a velocity sample is made at pixel 5 causing it to win. Just before 0.5s pixel 7 makes a velocity measurement and takes over the winning spot because pixel 5 is being inhibited by the inhibition of return circuitry.

INHIBITION OF RETURN

The output of the winner take all is used to drive the Inhibition of Return block, which takes the form of a second differential pair integrator (similar to the Slow Pulse block). The output of this block is an input to the Saliency Map which increasingly encourages the selection of a new pixel by the winner-take-all circuit depending on how long the current pixel has been selected for.

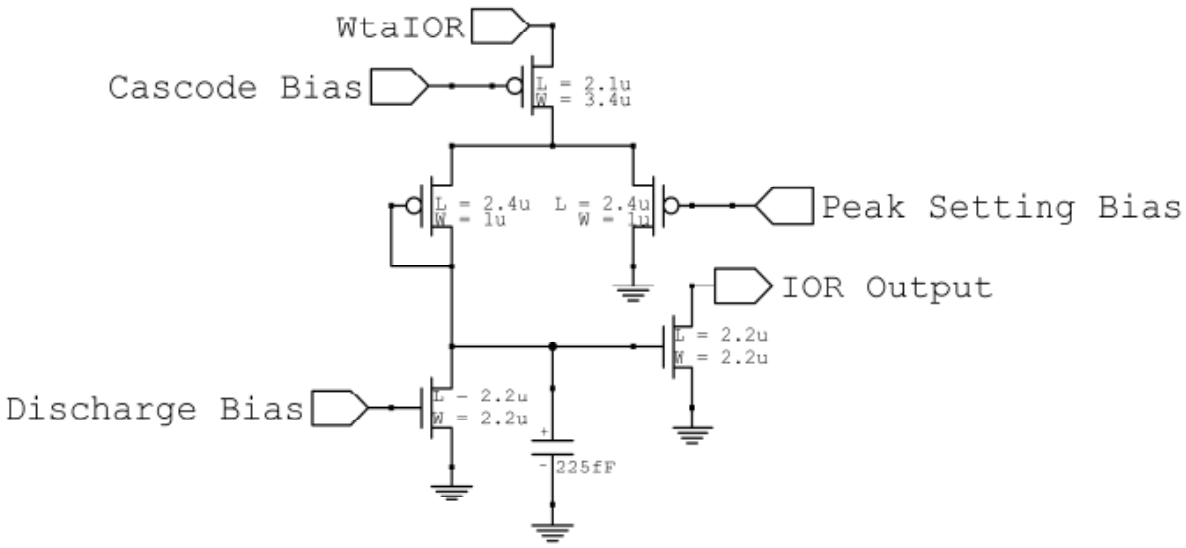

Figure 36: Inhibition of return circuit implemented as a differential pair integrator

SIMULATION

The Inhibition of Return DPI was simplified to increase the speed of simulation and therefore is not simulated in the same manner as the Slow Pulse DPI. The reason for the difference is that at each timestep of simulation the current winner of the winner take all must be determined before determining the next state of the IOR outputs. This slows down simulation considerably, so instead of modelling the complex charge and discharge profiles of the DPI, they are assumed to be linear.

The biases required for the desired peak voltage and charge/discharge times are still determined using the same equations as the Slow Pulse DPI. Transistor mismatch is taken into account in determining the actual rise and fall times.

The simulations show that the chip can accurately detect optical flow when transistor mismatch is ignored.

With the introduction of transistor mismatch of 20% the chip has a tendency to overestimate the optical flow by roughly 15% because when multiple pixels detect the optical flow it is biased to choose the largest value.

Simulation results so far appear to be in line with the results predicted by the equations for the high level simulation. Complexity continues to be added to the simulation framework in an attempt to more accurately approximate the operation of the chip.

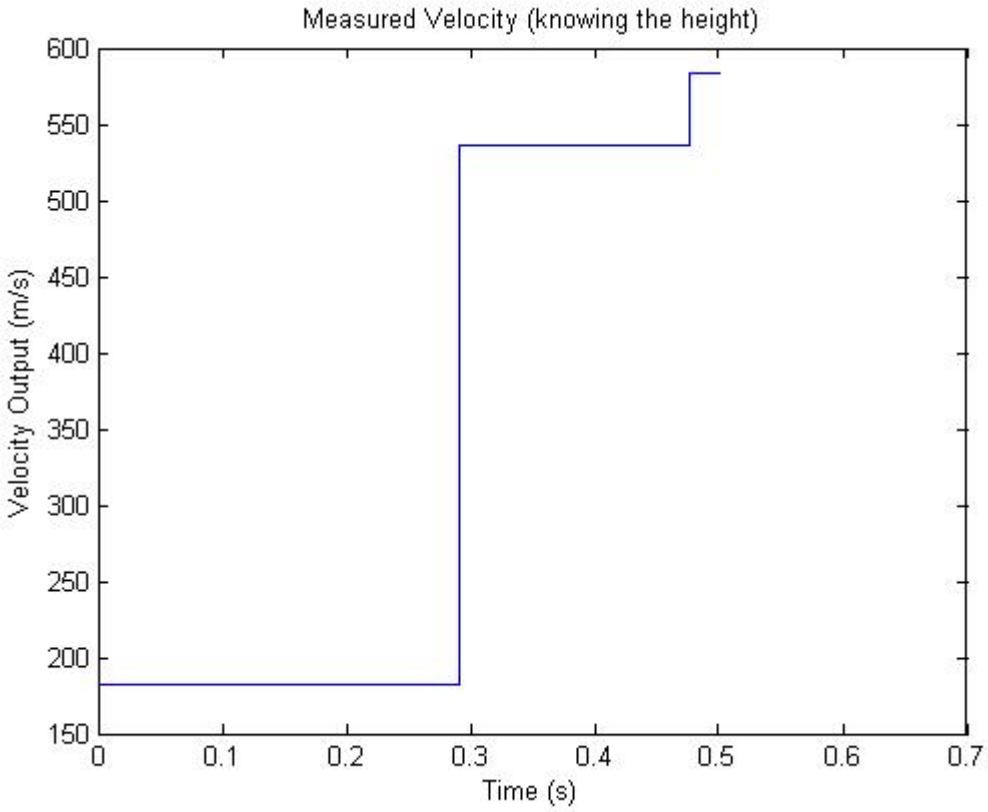

Figure 37: Velocity determined from the simulated chip output knowing the height from the planet surface. The actual velocity was 500m/s and is overestimated due to the non-zero array-velocity angle (20 degrees) and transistor mismatch effects.

In the above diagram there is initially no velocity reading. Just before 0.3s a reading is made on pixel 5 which wins the winner take all and therefore becomes the current output velocity. Just before 0.5s pixel 7 makes an even higher velocity measurement and becomes the new winner of the winner take all and therefore becomes the new output of the chip.

HIGH LEVEL SIMULATION

In the high level simulations we assume the use of two perpendicular sensing arrays. First we determine the actual optical flow seen by the sensor as a function of the sensor's location orientation, velocity (rotational and linear) and the intrinsic sensor parameters.

We then introduce error into the measurement based on observations of our low-level chip simulations

The final output of the sensor is given in units of pixels per second.

To determine the optical flow at a given pixel position requires knowledge of 3 things.

1) The z-component (in sensor co-ordinates) of the point on the surface which projects to the current pixel.

2) The translational velocity of the sensor.

3) The rotational velocity of the sensor.

Using knowledge of the above the optical flow can be computed. How these values are related to optical flow is shown later. The z-component of the point on the planet surface which projects to the current pixel can be extracted by finding the point of intersection of the surface with the line of points which project to the current pixel. The craft is considered rigid and therefore the velocity, position and orientation of the sensor can be inferred from the velocity, orientation and position of the craft.

Once the optical flow is known, the sensor output can be determined as a function of the actual optical flow, the orientation of the array and the orientation of edges in the image.

CO-ORDINATE SYSTEM DEFINITIONS

To find the optical flow we define four co-ordinate systems.

WORLD CO-ORDINATES

World co-ordinates are fixed relative to the planet surface with the planet surface lying on the x-y plane and the z-axis pointing into space.

Notation $p_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$

CRAFT CO-ORDINATES

Craft co-ordinates are fixed relative to the craft with the origin at the center of mass of the craft and the z-axis pointing towards the top of the craft.

Notation $p_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$

SENSOR CO-ORDINATES

Sensor co-ordinates are fixed to the sensor with the x-axis lying along one array and the y-axis along the other. The z-axis points outwards in the direction of viewing.

Notation $p_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix}$

PIXEL CO-ORDINATES

Pixel co-ordinates are only two dimensional and are simply a scaled version of the image plane with the x-axis and y-axis aligned with the sensor co-ordinates.

Notation $p_{pix} = \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix}$

The planet is simply modelled as a sphere with radius r and centre at $\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -r \\ 1 \end{bmatrix}$

In world co-ordinates the equation is simply

$$\begin{bmatrix} x_w \\ y_w \\ (z_w - r) \\ 1 \end{bmatrix}^T \begin{bmatrix} x_w \\ y_w \\ (z_w - r) \\ 1 \end{bmatrix} = r^2 + 1$$

This ensures that the origin of the world co-ordinates lies on the surface of the planet or moon with the z axis pointing away from the centre.

TRANSFORMING POSITION FROM WORLD TO SENSOR TO CRAFT CO-ORDINATES

We use Euler angles to define a rotation matrix, we then combine this with a translation to create a homogenous transformation from world to sensor co-ordinates.

The homogenous matrix H is the product of two intermediate homogenous matrices. First from world to craft co-ordinates and then from craft to sensor co-ordinates.

ROTATION

In this case we use a z-x-z rotation which is defined by three angles α, β and γ.

α is the angle between the old x axis and the line of intersection of the old and new x-y planes.

γ is the angle between the new x axis and the line of intersection of the old and new x-y planes.

β is the angle between the old and new z-axes.

All angles are measured counterclockwise looking towards the origin.

$$R = \begin{bmatrix} & & & 0 \\ & R_\gamma R_\beta R_\alpha & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_\alpha = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_\beta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & \sin(\beta) \\ 0 & -\sin(\beta) & \cos(\beta) \end{bmatrix}$$

$$R_\gamma = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos(\alpha)\cos(\gamma) - \sin(\alpha)\cos(\beta)\sin(\gamma) & \sin(\alpha)\cos(\gamma) + \cos(\alpha)\cos(\beta)\sin(\gamma) & \sin(\beta)\sin(\gamma) & 0 \\ -\cos(\alpha)\sin(\gamma) - \sin(\alpha)\cos(\beta)\cos(\gamma) & -\sin(\alpha)\sin(\gamma) + \cos(\alpha)\cos(\beta)\cos(\gamma) & \sin(\beta)\cos(\gamma) & 0 \\ \sin(\beta)\sin(\alpha) & -\sin(\beta)\cos(\alpha) & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

TRANSLATION

The translation matrix from one co-ordinate frame to another is simply

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $[x_0, y_0, z_0]$ is the location of the origin of the new set of axis in the old co-ordinate frame.

THE TRANSFORMATION MATRIX

A homogenous transformation matrix can be made by multiplying the rotation and translation matrix.

$$H_i = R_i T_i$$

The inverse matrix can be obtained using the

$$H_i^{-1} = T_{i_{reverse}} R_i^T$$

where

$$T_{i_{reverse}} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

H_1 is the transformation from world to craft co-ordinates and H_2 is the transformation from craft to sensor co-ordinates. This gives the transformations

$$p_c = H_1 p_w$$

$$p_s = H_2 p_c$$

TRANSFORMING POSITION FROM SENSOR TO PIXEL CO-ORDINATES

In simulation the craft position, orientation and velocity are known in planet co-ordinates. Knowing this and the position and orientation of the sensor in the craft co-ordinate system we can determine the sensor velocity, orientation and position in planet co-ordinates. Any position in sensor co-ordinates can be transformed to pixel co-ordinates using the following transformation.

$$\begin{bmatrix} u_{pix} \\ v_{pix} \\ w_{pix} \end{bmatrix} = A_{pix} p_{sensor}$$

$$\begin{bmatrix} u_{pix} \\ v_{pix} \\ w_{pix} \end{bmatrix} = \begin{bmatrix} p_x & 0 & 0 & 0 \\ 0 & p_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix}$$

Where $p_x = p_y = \left(\frac{64}{2 \tan\left(\frac{FOV}{2}\right)} \right)$ and FOV is the field of view of the sensor.

The actual pixel values are

$$\lambda \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} u_{pix} \\ v_{pix} \\ w_{pix} \end{bmatrix}$$

Where lambda must be found such that the third value in the pixel vector is 1.

The inverse transformation is given by

$$\begin{bmatrix} \lambda x_s \\ \lambda y_s \\ \lambda z_s \\ 1 \end{bmatrix} = A_{sensor} p_{pix}$$

Where λ is now ambiguous, thus giving the equation of a line, originating at the origin not a point. λ must be positive as negative lambda will refer to points behind the camera.

A_{sensor} is given by

$$A_{sensor} = \begin{bmatrix} \frac{1}{p_x} & 0 & 0 \\ 0 & \frac{1}{p_y} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Note that this inverse transform describes a line, not a point because the depth of the point projecting to the pixel is ambiguous.

TRANSFORMING VELOCITY FROM CRAFT TO SENSOR CO-ORDINATES

Assuming a fixed planet surface there is no need to consider planet motion. The velocity of the craft in the world co-ordinates can easily be transformed to craft co-ordinates in the same manner as position was transformed. Transforming from craft co-ordinates to sensor co-ordinates is not as simple though because rotation of the craft can introduce both rotation and translation of the sensor. We know the velocity of the sensor in craft co-ordinates is.

$$V_c = R_2 V_w$$

If the craft is rotating with velocity Ω_c in craft co-ordinates it will cause the sensor to translate by an additional $\Omega_c \mathbf{x}\, p_c$ in the non-rotating craft frame where p_c is the position of the sensor in craft co-ordinates.

Thus

$$\Omega_s = R_1 \Omega_c$$

$$V_s = R_1(V_c + \Omega_c \mathbf{x}\, p_c)$$

$$= R_1(R_2 V_w + \Omega_c \mathbf{x}\, p_c)$$

We now have a complete representation for the translational and rotational velocity of the sensor.

FINDING THE Z-COMPONENT IN SENSOR CO-ORDINATES OF THE SURFACE POINT WHICH MAPS TO THE CURRENT PIXEL

To find the position of a point mapping to a pixel we first determine the equation of the line of points projecting to the current pixel. We then convert this line to world co-ordinates and find its point of intersection with the planet surface. Finally, we convert the point of intersection back into sensor co-ordinates and extract the z-component.

Given a pixel position $p_{pix} = \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix}$ we can find the equation of the line of points projecting to the pixel using the transformation

$$\begin{bmatrix} \lambda x_s \\ \lambda y_s \\ \lambda z_s \\ 1 \end{bmatrix} = A_{sensor} p_{pix}$$

We can then split the sensor co-ordinate into two terms

$$\begin{bmatrix} \lambda x_s \\ \lambda y_s \\ \lambda z_s \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} x_s \\ y_s \\ z_s \\ 0 \end{bmatrix}$$

Considering each term individually we can find the equation of a line in world co-ordinates

We find the origin of the line by

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = H_2^{-1} H_1^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

And the direction of the line by

$$\lambda \begin{bmatrix} \left(\frac{dx}{d\lambda}\right) \\ \left(\frac{dy}{d\lambda}\right) \\ \left(\frac{dz}{d\lambda}\right) \\ 0 \end{bmatrix} = \lambda H_2^{-1} H_1^{-1} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 0 \end{bmatrix}$$

To find the point of intersection of this line with the surface we find a value λ such that

$$\left(\begin{bmatrix} x_0 \\ y_0 \\ z_0 + r \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} \left(\frac{dx}{d\lambda}\right) \\ \left(\frac{dy}{d\lambda}\right) \\ \left(\frac{dz}{d\lambda}\right) \\ 0 \end{bmatrix}\right)^T \left(\begin{bmatrix} x_0 \\ y_0 \\ z_0 + r \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} \left(\frac{dx}{d\lambda}\right) \\ \left(\frac{dy}{d\lambda}\right) \\ \left(\frac{dz}{d\lambda}\right) \\ 0 \end{bmatrix}\right) = r^2 + 1$$

We can rewrite the equation for the point of intersection as:

$$\lambda^2\left(\left(\frac{dx}{d\lambda}\right)^2 + \left(\frac{dy}{d\lambda}\right)^2 + \left(\frac{dz}{d\lambda}\right)^2\right) + \lambda\left(2x_0\frac{dx}{d\lambda} + 2y_0\frac{dy}{d\lambda} + (2z_0 + 2r)\frac{dz}{d\lambda}\right) + (x_0^2 + y_0^2 + z_0^2 + 2z_0 r) = 0$$

And solve for λ using the formula for the roots of a quadratic equation. We constrain λ to being real and positive. If two such values exist we use the lower value, indicating where the line first intersects the sphere. We can then map this point back to sensor co-ordinates using

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} = H_1 H_2 \left(\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} \left(\frac{dx}{d\lambda}\right) \\ \left(\frac{dy}{d\lambda}\right) \\ \left(\frac{dz}{d\lambda}\right) \\ 0 \end{bmatrix}\right)$$

This giving us the value z_s

Using this information we can describe the optical flow due to translation as:

$$\frac{dx_{pix}}{dt} = \frac{V_{z_s} x_{pix} - V_{x_s}}{z_s}$$

$$\frac{dy_{pix}}{dt} = \frac{V_{z_s} y_{pix} - V_{y_s}}{z_s}$$

If the sensor is rotating such that the rotation is described by the matrix S where

$$\frac{dp}{dt} = S \, \mathbf{x} \, p$$

Then the optical flow due to this rotation is:

$$\frac{dx_{pix}}{dt} = -w_y + w_z y_{pix} + w_x \frac{x_{pix} y_{pix}}{\sqrt{p_x p_y}} - \frac{w_y x_{pix}^2}{p_x}$$

$$\frac{dy_{pix}}{dt} = w_x - w_z x_{pix} - \frac{w_y x_{pix} y_{pix}}{\sqrt{p_x p_y}} + \frac{w_x y_{pix}^2}{p_y}$$

Where p_x and p_y are as defined in the A_{pix} matrix.

Thus we get a final optical flow

$$\frac{dx_{pix}}{dt} = \frac{V_{z_s} x_{pix} - V_{x_s}}{z_s} - w_y + w_z y_{pix} + w_x \frac{x_{pix} y_{pix}}{\sqrt{p_x p_y}} - \frac{w_y x_{pix}^2}{p_x}$$

$$\frac{dy_{pix}}{dt} = \frac{V_{z_s} y_{pix} - V_{y_s}}{z_s} + w_x - w_z x_{pix} - \frac{w_y x_{pix} y_{pix}}{\sqrt{p_x p_y}} + \frac{w_x y_{pix}^2}{p_y}$$

We define the optical flow v_{image} in pixel co-ordinates as

$$v_{image} = \begin{bmatrix} \frac{dx_{pix}}{dt} \\ \frac{dy_{pix}}{dt} \end{bmatrix}$$

OPTICAL FLOW DETECTED BY THE SENSOR

The detected optical flow in the direction of a linear array is determined by three factors.

1) The angle between the edge and the direction of optical flow θ_1
2) The angle between the array and the direction of optical flow θ_2
3) The speed at which the edge travels across the image plane v_{image}

The optical flow in the direction of the array is then given by the following equation

$$\frac{v_{image} \sin \theta_1}{\sin(\theta_1 - \theta_2)}$$

The second array in a sensor will then give a reading

$$\frac{v_{image} \sin \theta_1}{-\cos(\theta_1 - \theta_2)}$$

θ_2 and v_{image} can be determined using the axes transformation described previously. The probability of encountering an edge with angle θ_1 to the motion can be approximated by assuming that the edges are crater boundaries and therefore circular. This results in a probability distribution of:

$$p(\theta_1) = \frac{\sin \theta_1}{2}$$

This is only true for a long continuous edge. For short edges we can assume that the ends of the edge lie within the field of view. In this case it is as if the edge lies perpendicular to the direction of motion ($\theta_1 = 90^0$).

Using this probabilistic approach the sensor outputs can be determined under ideal conditions. In practice error is introduced in the measurement process. The two main causes of this are transistor mismatch and quantisation error introduced when reading out results.

We are working to confirm these expected results with low level simulations.

PREDICTION OF SIMPLIFIED HIGH LEVEL SIMULATIONS COMPARED TO LOW LEVEL SIMULATIONS

As discussed on the previous page, the relative direction of the array, velocity and edge all affect the measurement. For the sake of simplicity we ignore the effect of edge orientation. The predicted output and the simulated output are compared below. For these simulations the transistor mismatch noise was fixed across simulations.

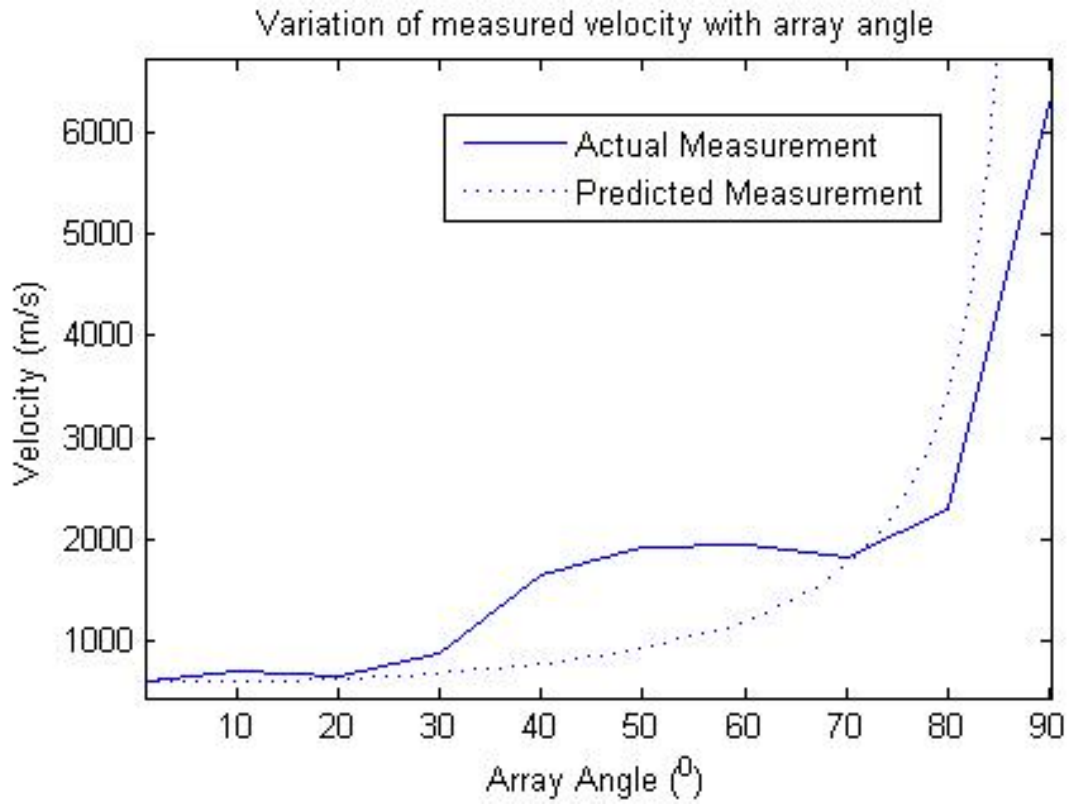

Figure 38: Error introduced into the velocity measurement as a function of the angle between the array and the velocity.

For an angle of zero degrees the measurement is very accurate, but the error increases exponentially as the array to velocity angle increases. The difference between the measured and predicted results can be attributed to the effects of the edge angle which have been ignored in high level simulations. The edge angle also varies across simulations in the above plot because as the array is rotated, different sections of the array will see different edges. Error introduced by transistor mismatch also plays a small role.

HIGH LEVEL SIMULATION OUTPUTS

Some of the simpler more intuitive examples from the high level simulation are shown below. The simulation is by no means restricted to these simple cases. The simulation can handle an arbitrary combination of rotational and translational velocities.

APPROACHING OPTICAL FLOW

The following plots were generated using purely translational optical flow

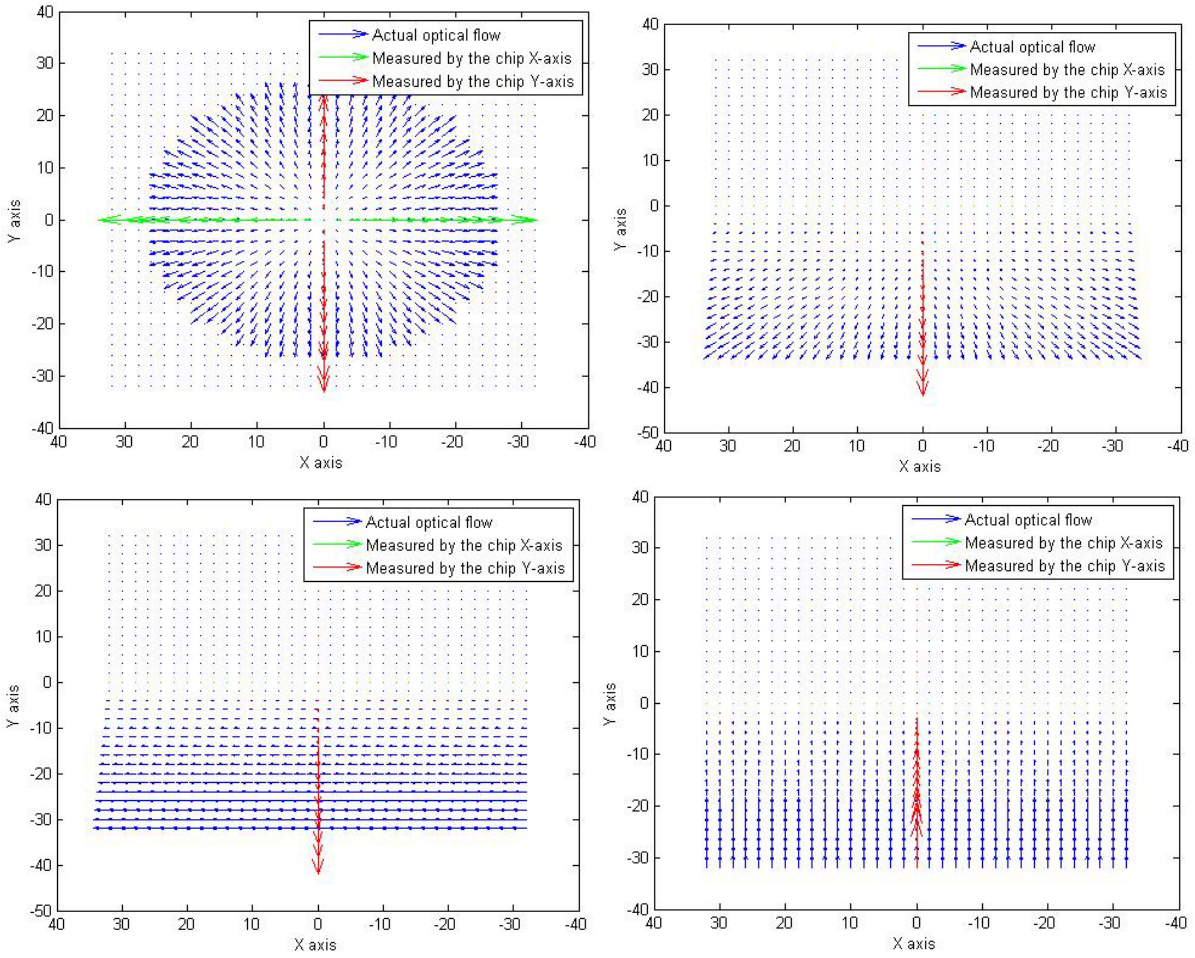

Figure 39: High Level simulation optical flow output when approaching a planet from afar (top left) and when close (top right) flying just above the surface of the planet. The bottom two are generated with the same sensor position as the top right but with velocities to the right (bottom left) and downwards (bottom right)

In the bottom left plot the large red arrows appear erroneous as there should be no vertical optical flow, but as discussed on the previous page, when optical flow is perpendicular to the array it will output a very large velocity reading. The plotting function in Matlab scales the arrows to make them easier to read, but the arrows in the other three plots represent much smaller magnitudes than the red arrows of the bottom left.

ROTATION

The simulation outputs below show optical flow for rotations about the sensor x, y and z axis. In this case the sensor is looking directly down at the planet.

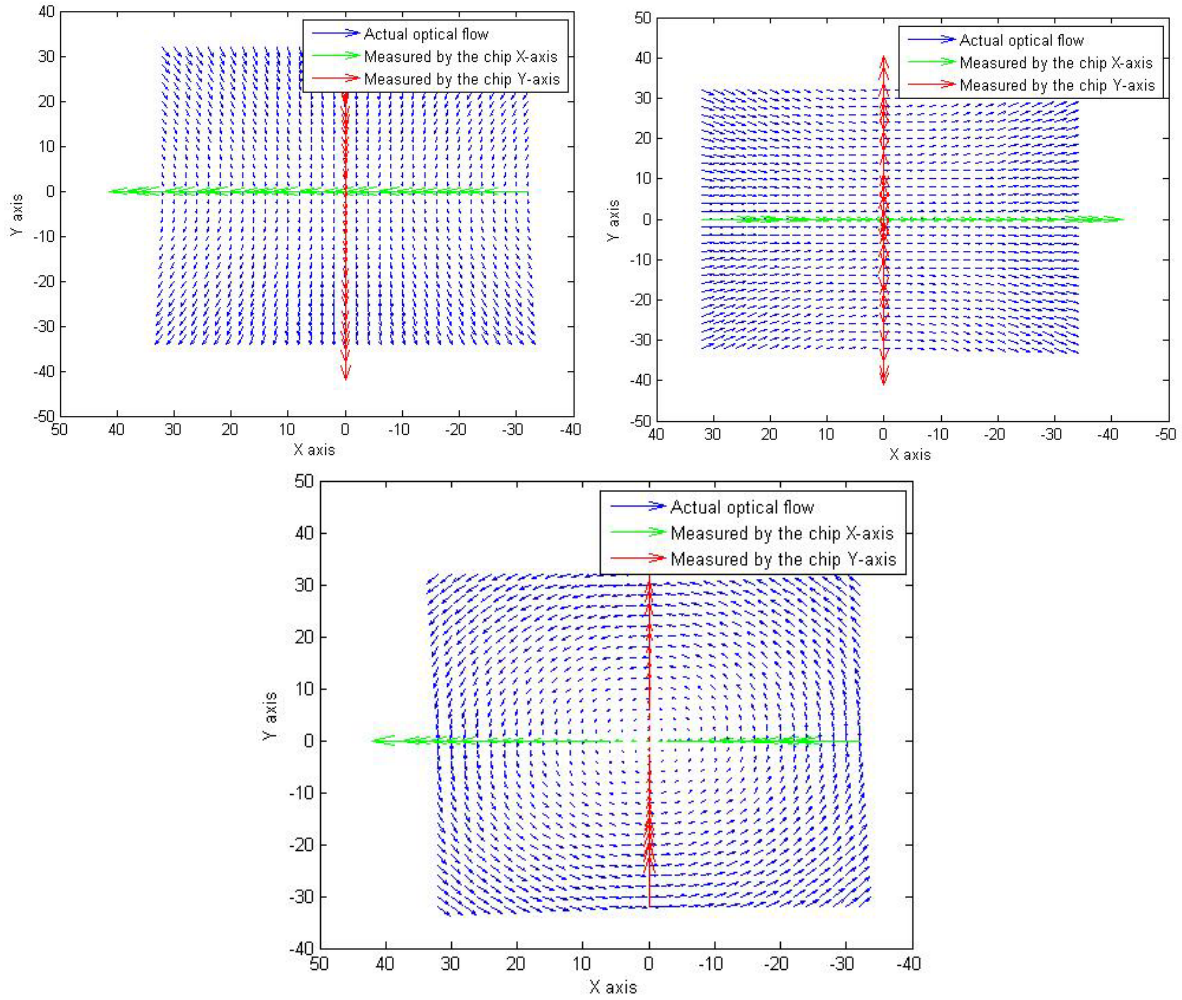

Figure 40: Simulation ouput for rotations about the sensor x-axis (top left) y-axis (top right) and z-axis (bottom)

CONCLUSIONS

TRANSISTOR LEVEL SIMULATIONS

The photoreceptor does not respond well to high frequencies, but this should not be an issue in the current application where only low frequency signals are expected.

In transistor level simulations the temporal differentiator circuit was prone to oscillate. It is unclear whether this is due to error in simulation methods. Chip measurements which are expected shortly will clarify this.

The spatial derivative is computed using a bump-antibump circuit which is very sensitive to transistor mismatch and is unlikely to be of use in small processes which are prone to mismatch.

FULL CHIP LOW LEVEL SIMULATIONS

With transistor mismatch of up to 20% the full chip low level simulation suggest that the velocity will be overestimated by approximately 15% when the optical flow is in the direction of the array.

When optical flow has a non-zero angle to the array the velocity reading increases non-linearly with the angle.

Pre-calculated biases must be double checked to ensure that they do not invoke above threshold operation of transistors.

Chip simulations greater than 0.5s cause Matlab to run out of memory.

The chip output is also dependant on the angle of the edge in an image, but this cannot be determined by examining the image itself.

FULL CHIP HIGH LEVEL SIMULATIONS

The high level chip simulations provide outputs by determining the actual expected optical flow and then taking into account transistor mismatch and the effect of optical flow direction from low level simulations.

In order to accurately determine optical flow using a single chip a 2 dimensional sensing array is required.

REFERENCES

[1] Shih-Chii Liu, Jörg Kramer, Giacomo Indiveri, Tobias Delbrück, Rodney Douglas, "Analog VLSI: Circuits and Principles" MIT Press, 2002

[2] Bartolozzi, C.; Mitra, S.; Indiveri, G., "An ultra low power current-mode filter for neuromorphic systems and biomedical signal processing," *Biomedical Circuits and Systems Conference, 2006. BioCAS 2006. IEEE* , vol., no., pp.130-133, Nov. 29 2006-Dec. 1 2006

[3] S. Liu, "Silicon retina with adaptive filtering properties," in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.). Cambridge, MA: MIT Press, 1998, vol. 10

[4] Indiveri, G. "A Current-mode Analog Hysteretic Winner-take-all Network, with Excitatory and Inhibitory Coupling", *Analog Integrated Circuits and Signal Processing*, 28:(3) 279-291, Sep, 2001