Centeye **Stonyman / Hawksbill Silicon Documentation**

Stonyman and Hawksbill vision chips

Revision 1.0

March 12, 2013

1. General Description:

The Stonyman and Hawksbill chips are minimalist image sensor chips designed for a broad set of visual sensing applications. Currently two chips are available in this series: The Stonyman chip has a square pixel array and a resolution of 112 x 112 pixels, while the Hawksbill chip uses a hexagonal pixel array and has a resolution of 136 x 136 pixels. The Stonyman chip additionally has binning in the focal plane allowing blocks of M x N pixels (M and N independently selectable from 1, 2, 4, or 8) to form square or rectangular super pixels.

Both chip use a simple interface requiring just five digital inputs and providing one analog output. The chips are operated by pulsing the five digital inputs according to an intuitive sequence described below. With the two power rails GND and VDD, this allows each chip to be connected to a circuit using just eight connections (or wire bonds). An additional optional “OUTENABLE” signal, internally pulled up, allows multiple chips to share the same analog output line.

Both chips use continuous-time logarithmic pixels, thus allowing operation over a large range of image intensities.

For advanced users, both chips contain an analog pre-amplifier allowing individual pixel signals to be amplified before being sent out. The gain of this amplifier is selectable from seven levels, and the offset is selectable from 64 levels.

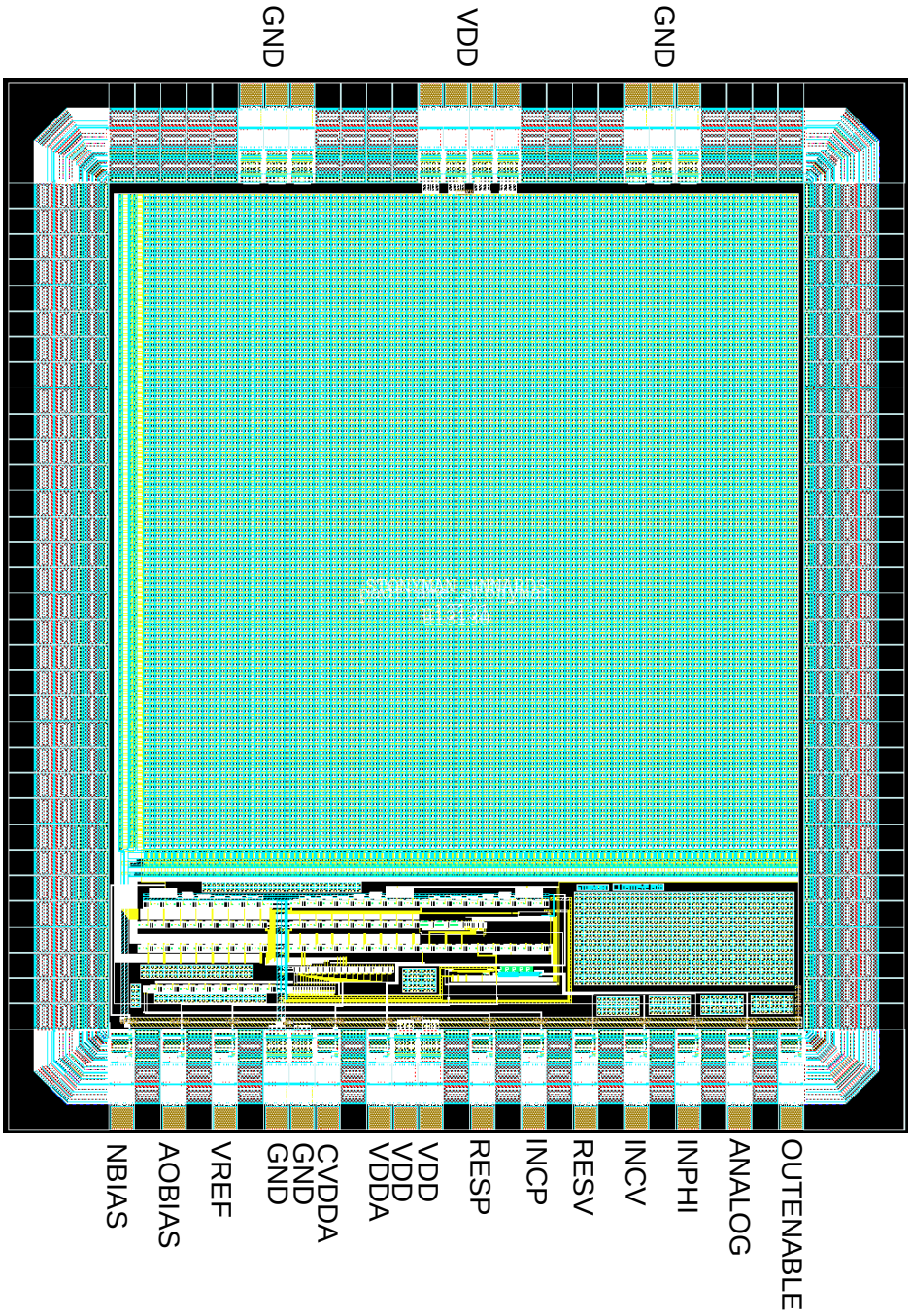
2. General Specifications:

Raw focal plane resolution	Stonyman: 112x112 (12,544) Hawksbill: 136x136 (18,496)	
Array type	Stonyman: Square array Hawksbill: Hawksbill: Hexagonal array arranged in ASA Form 0	
Drawn die size	Stonyman: 4535um x 3665um Hawksbill: 4095um x 3665um	Actual die size will be slightly larger due to cutting of reticle
Focal plane size	Stonyman: 2.8mm x 2.8mm Hawksbill: 2.3mm x 32.7mm	
Pixel pitch	Stonyman: 25 microns Hawksbill: 20 microns	"pitch" is distance between centroids of light sensing regions. For Stonyman this is the length of one edge as defined by a 2-by-2 block of pixels. For Hawksbill this is the length of one edge as defined by a triangle of three pixels.
Pixel circuitry	Logarithmic / continuous time pixel	
Pixel binning	Stonyman: in-pixel connections to N, S, E, and W pixels Hawksbill: None	
Pixel amplifier	Seven-level switched capacitor circuit	May be bypassed to allow raw pixels to be read out
Interface	Counter based, with five digital inputs and one analog output	
Process	X-Fab XC06	
Power supply voltage	VDD=3.0V to 5.0V	Operation below about 4.0V requires the use of the amplifier to implement level shifting

3. Chip Layout and Bonding Information:

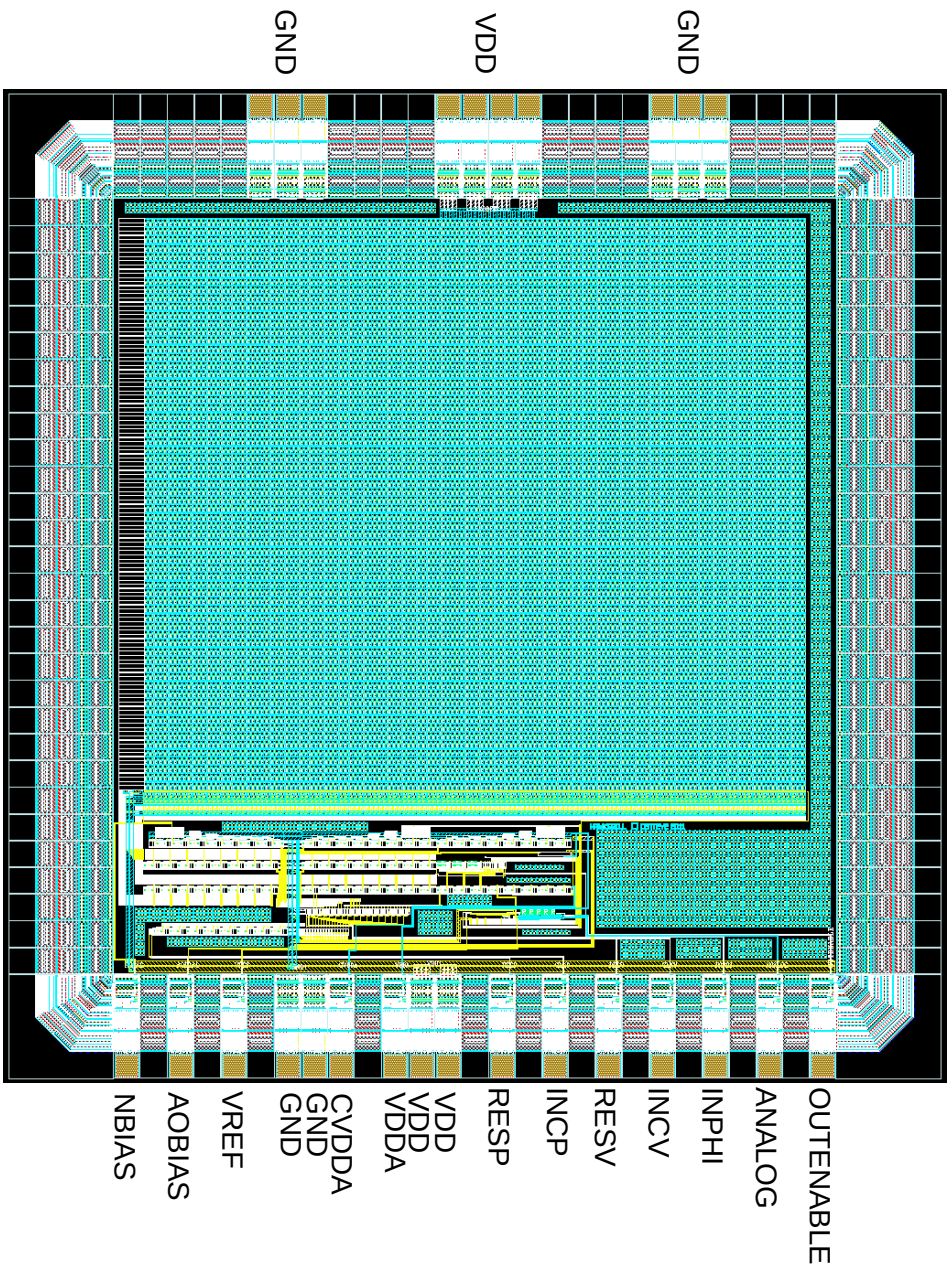
The following pages show the layout of the Stonyman and Hawksbill chips. The pad frame is identical for both chips, except for the fact that Stonyman is slightly taller.

3.1 Stonyman



Stonyman

3.2 Hawksbill



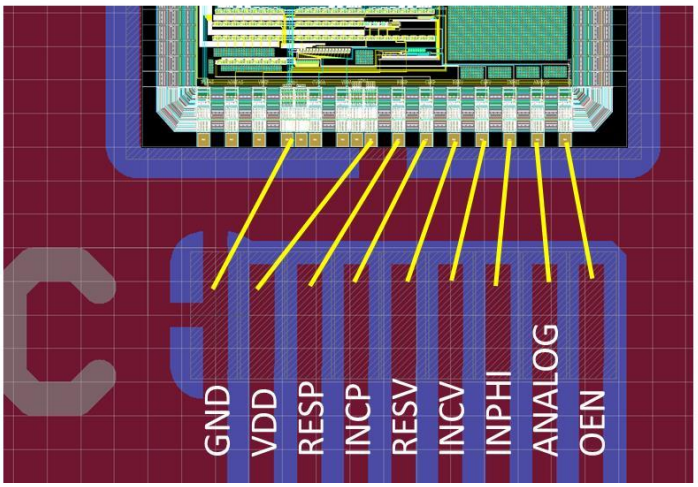
Hawksbill

3.3 Wire Bonding and Encapsulation

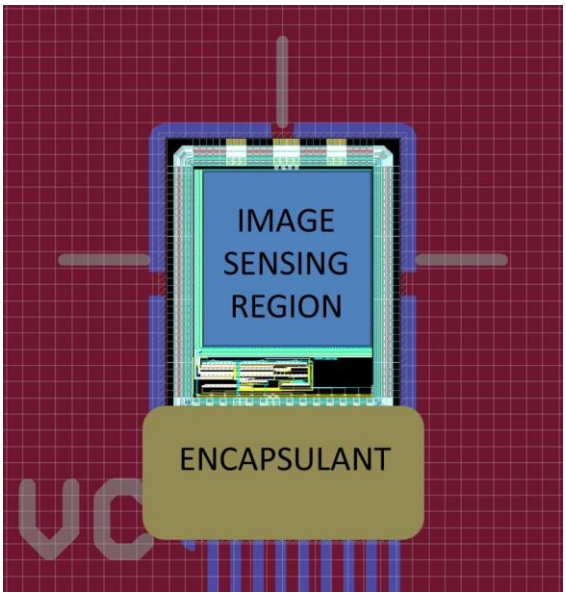
This section is for customers acquiring these chips in bare die form. If you have acquired these chips on a breakout board or other PCB, then you may skip this section.

The pads and the spacers between pads have a pitch of 110 microns. For example, the distance between the centers of the NBIAS and AOBIAS pads is 220 microns, while the distance between the centers of adjacent GND pads (or the centers of adjacent VDD pads) is 110 microns. The bonding pads themselves are about 85 microns in width and height.

The picture below shows a sample minimal wire bonding diagram that provides access to the most important signals of these chips. The power and ground pads at the top of the chip may be used as well to improve these power connections, but this is not required.



For encapsulation, it is important that only the wire bonds are encapsulated. If encapsulant spills over onto the focal plane, then some of the pixels may be rendered useless.



4. List of pads / external signals:

Below is a list of all pads and external signals which is valid for all Stonyman and Hawksbill series chips. All pads having the same name are electrically connected on the chip. Most users will need to use only the subset of signals that are "black". Advanced and optional signals are in "blue" text. Most users will not need these signals, with the exception of the OUTENABLE signal when multiple chips are connected to the same analog output line.

Name	Type	Description
VDD	Power	Power supply to all portions of the chip except bias generators
GND	Power	Ground/substrate
RESP	Digital Input	Reset Pointer. Resets pointer register when pulsed. Active high.
INCP	Digital Input	Increment Pointer. Increments pointer register when pulsed. Active high.
RESV	Digital Input	Reset Value. Resets the active register (as selected by pointer register). Active high.
INCV	Digital Input	Increment Value. Increments the active register (as selected by pointer register). Active high.
INPHI	Digital Input	Operates amplifier. Rest state is low. Raise to reset amplifier, lower to generate amplified output.
ANALOG	Analog Output	Analog output of chip containing pixel signal. This gets sent to an ADC.
NBIAS	Bias	Supply for column readout circuits
AOBIAS	Bias	Bias for final output buffer amplifier
VREF	Bias	Reference voltage for amplifier circuit
VDDA	Bias generator power	Power signal for bias generators. Pulled down to GND on power up, may be connected to VDD using the CONFIG system register. When the chip is turned on using the CONFIG register (see Section 8.2) this pad is set to VDD.
CVDDA	Analog Power Control	Determines whether VDDA and VDD are shorted. When this signal is VDD, lines VDD and VDDA are disconnected. When this signal is GND, lines VDD and VDDA are connected. This signal is VDD on power up. Turning on the chip with the CONFIG system register (see Section 8.2) sets this pad to GND potential.
OUTENABLE	Digital Input	Functions as a “chip select” line. Connects output buffer amplifier to ANALOG pin when digital high. Internally pulled up to VDD. Pull down to GND to disconnect chip output.

5. Connecting a Stonyman or Hawksbill chip to power and to a processor:

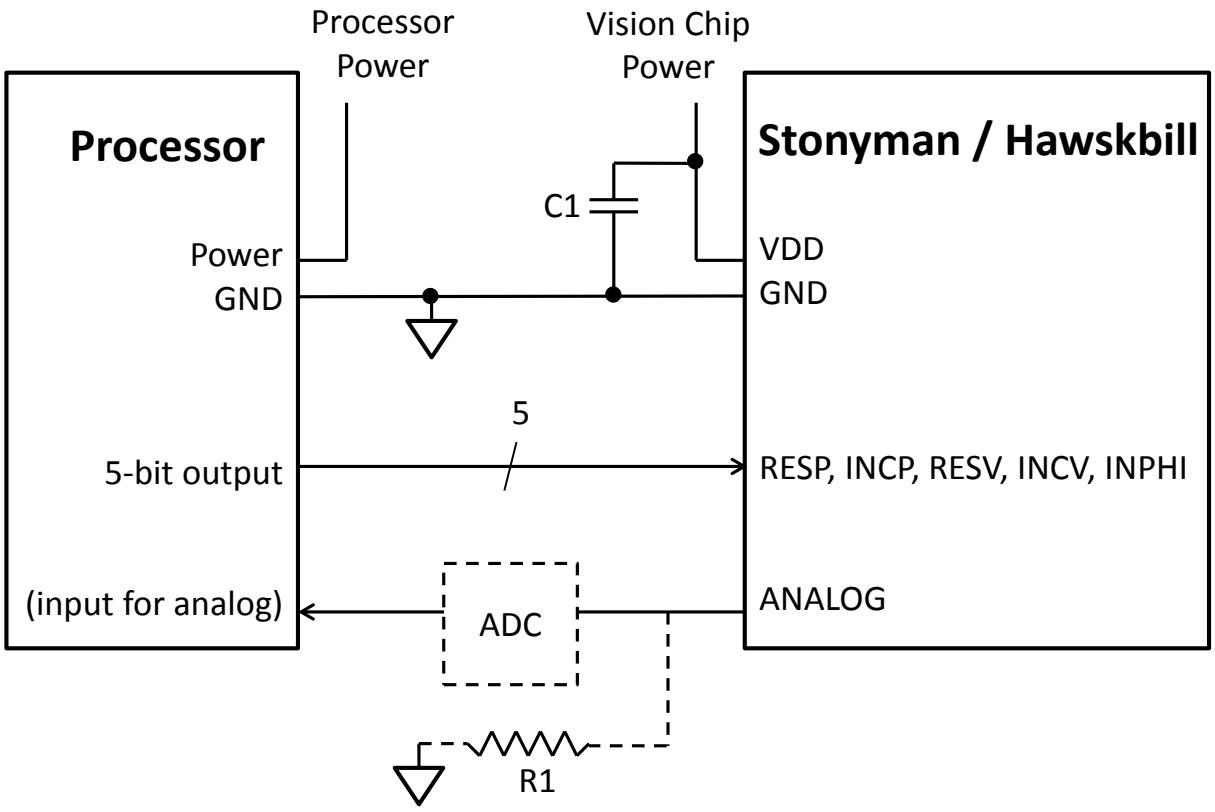
The figure below shows a minimal connection between a Stonyman/Hawksbill vision chip and a generic processor operating it. The interface requires five digital lines going from the processor to the vision chip. The vision chip outputs one signal which gets sent to the processor. This analog signal must be digitized, either using an external ADC or one located within the processor.

The single best way to obtain optimal performance from the vision chip is through clean power supplies. This includes a clean power supply to the vision chip as well as to any ADC that is used. If the ADC is located on the processor, then care should be taken to ensure that the analog power supply to the processor that powers the ADC is also clean. Note that it is possible for the processor and the vision chip to be powered by the same line. In this case, adequate bypass capacitors should be used to minimize any "glitches" that may be sent to the vision chip.

The vision chip and the processor may be powered with a single power supply or by separate power supplies. It is a requirement that any digital signals sent to the vision chip not exceed VDD in potential.

Bypass capacitors: It is recommended that one or more bypass capacitors (e.g. C1) be placed across the power rails of the vision chip. A 1nF capacitor and optionally a 0.1uF capacitor are suggested.

ANALOG pull-down resistor: It is suggested that a pull down resistor (e.g. R1) be placed between the ANALOG signal and ground as shown below. This is not required. A suggested value for R1 is 5k to 20k. This resistor may be used to decrease the output impedance of the vision chip output, however may result in an attenuation of the output voltage.



Note: GND < RESP,INCP,RESV,INCV,INPHI ≤ VDD

6. Commands:

The Stonyman and Hawksbill vision chips are operated using commands sent to the chip using the input signals RESP, INCP, RESV, and INCV. This is discussed further in Section 8.1. Input signal INPHI is used to operate the pixel amplifier, also described below.

7. Chip Block Diagram:

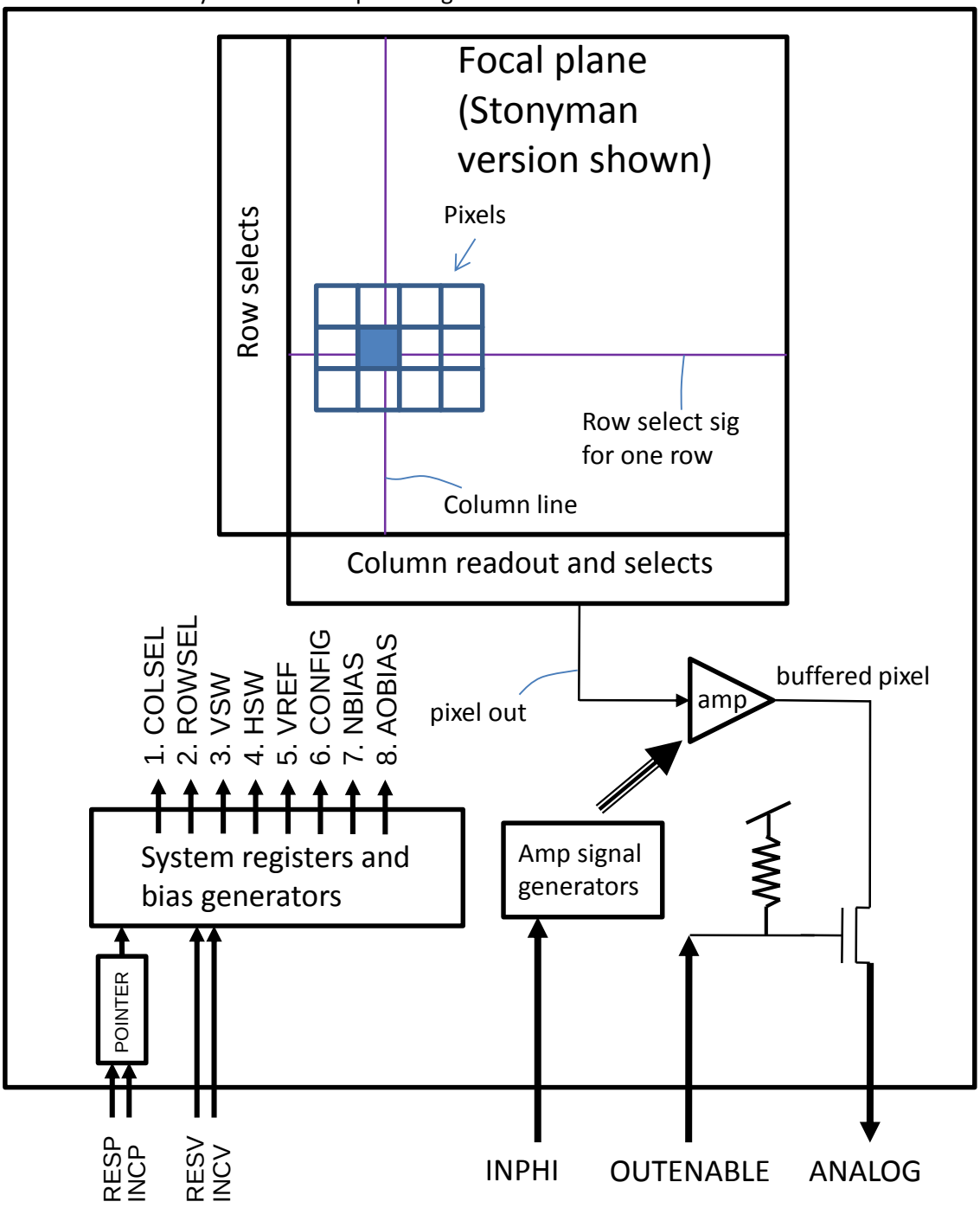
7.1 Overview

The figure below shows a high level block diagram of the chip. The eight system registers direct the operation of the chip including all configuration information and bias levels. This includes the selection of row and column for pixel readout. The system registers are operated using an "increment or reset" scheme, where each register can either be incremented or reset. The system registers are configured using the four digital inputs RESP, INCP, RESV, and INCV. See Section 8.1 for further details on how to configure system registers.

The focal plane contains the pixel circuits themselves. For the Stonyman chip, the focal plane also contains binning circuitry. The system registers ROWSEL and COLSEL select the pixel to be read out. A single amplifier amplifies the pixel signal and provides a buffered output to the ANALOG output pad

which may be sent to an external ADC. The INPHI input signal, along with the CONFIG and VREF system registers, affect the operation of the amplifier.

An additional input signal OUTENABLE allows multiple chips to have their analog outputs connected to the same line. When OUTENABLE is a digital high, that chip is “selected” and provides an output to the ANALOG pad. The OUTENABLE signal is internally pulled high therefore it may be left disconnected if only one vision chip is being used.



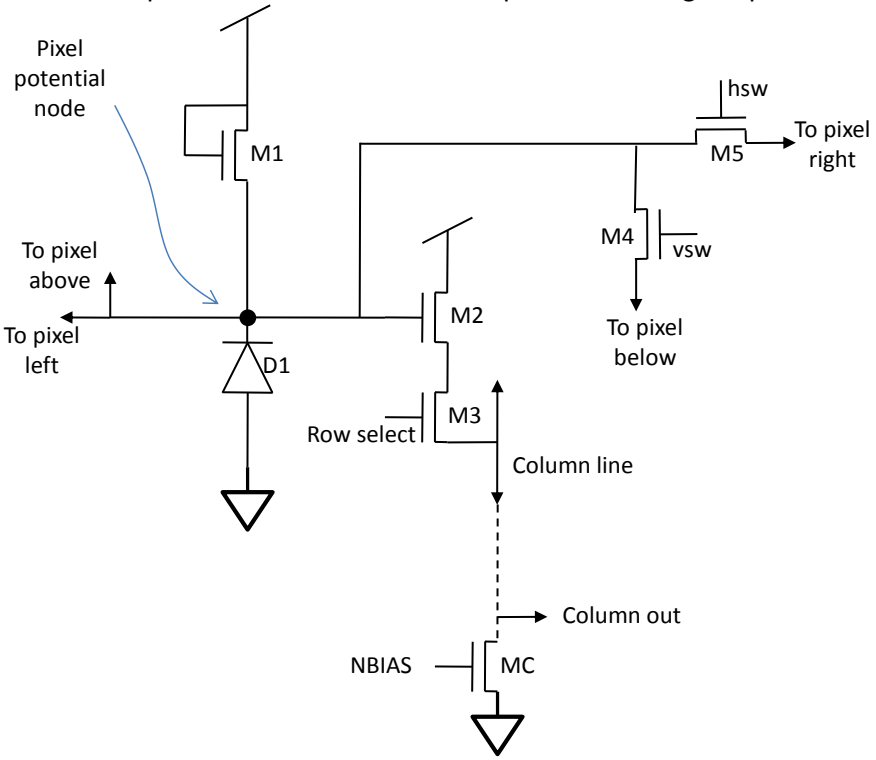
7.2 Pixel Circuits:

7.2.1 Stonyman pixel circuit

The pixel circuit for the Stonyman chip is shown below. The light sensing element is N-well photodiode D1 (formed between an N-well and the P-substrate, the latter tied to ground). The function of M1 will be discussed below. The voltage generated by the pixel circuit is stored at the node with the circle (•). When the row select signal is high, the voltage is read out to the column line through transistors M2 and M3. The column line goes to transistor MC, which with the bias NBIAS and transistor M2 forms a source follower to output the pixel value. Note that there is only one MC transistor for each column line.

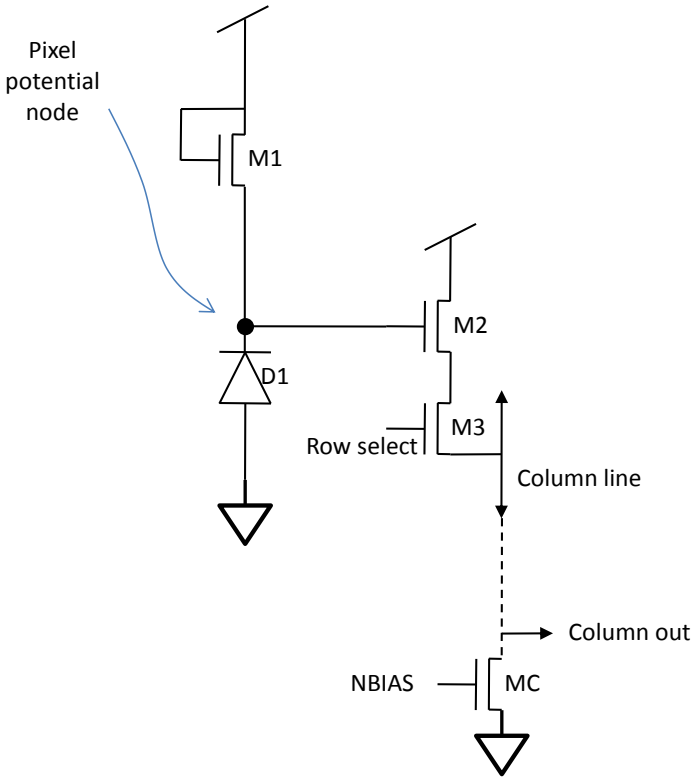
As light strikes the photodiode D1, it sinks current to the substrate (to ground) proportional to the amount of light striking it. This current is typically in the picoamp or nanoamp range. This small current flows through transistor M1. Due to the small current, this transistor operates in the subthreshold region. When diode connected, the voltage drop across M1 is a logarithmic function of the current flowing through them, and thus a logarithmic function of the intensity. When light changes by a factor of e=2.7, the voltage drop across M1 will change by about 50mV to 80mV. A large range of light intensities may thus be compressed within a manageable voltage swing.

The signals vsw and hsw may be used to electrically connect a pixel circuit to the pixel circuits above or to the right. This allows groups of pixels to be "binned" together to form "superpixels" covering a larger portion of the focal plane. Instructions on how to perform binning are provided below.



7.2.2 Hawksbill pixel circuit

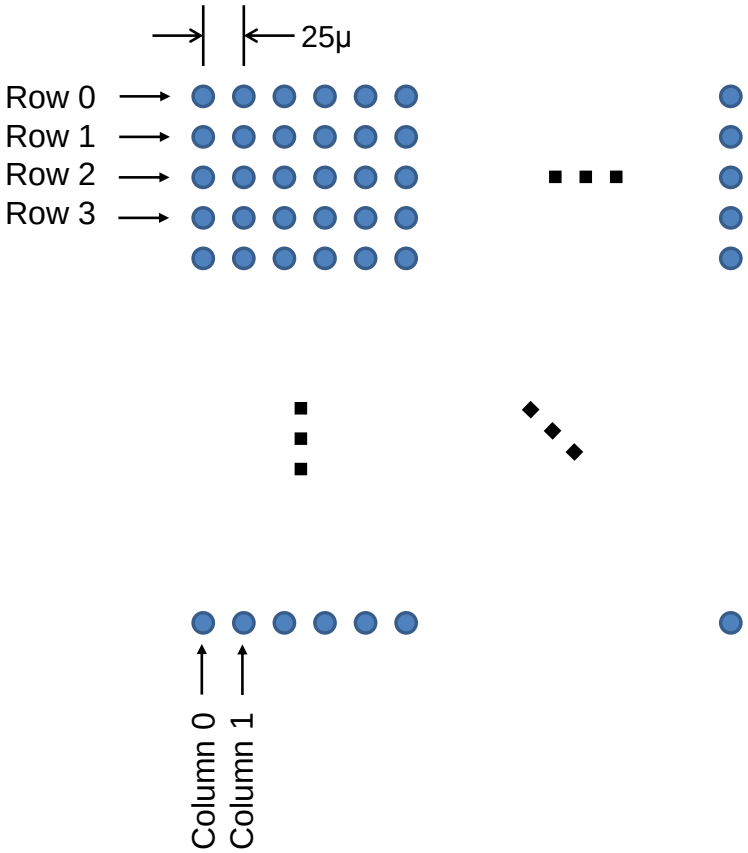
The Hawksbill pixel circuit is shown below. The Hawksbill pixel circuit is identical to the Stonyman pixel circuit except that binning transistors M4 and M5 are not used. Therefore a Hawksbill vision chip does not support pixel binning.



7.3 Focal Plane Structure

7.3.1 Stonyman focal plane

The Stonyman vision chip has a resolution of 112 x 112 pixels, with row 0, column 0 being the top left of the chip (as shown in Figure 3). The pitch between pixels, which is defined as the distance between the centers of two horizontally or vertically adjacent pixels, is 25 microns. The figure below shows the organization of the focal plane.

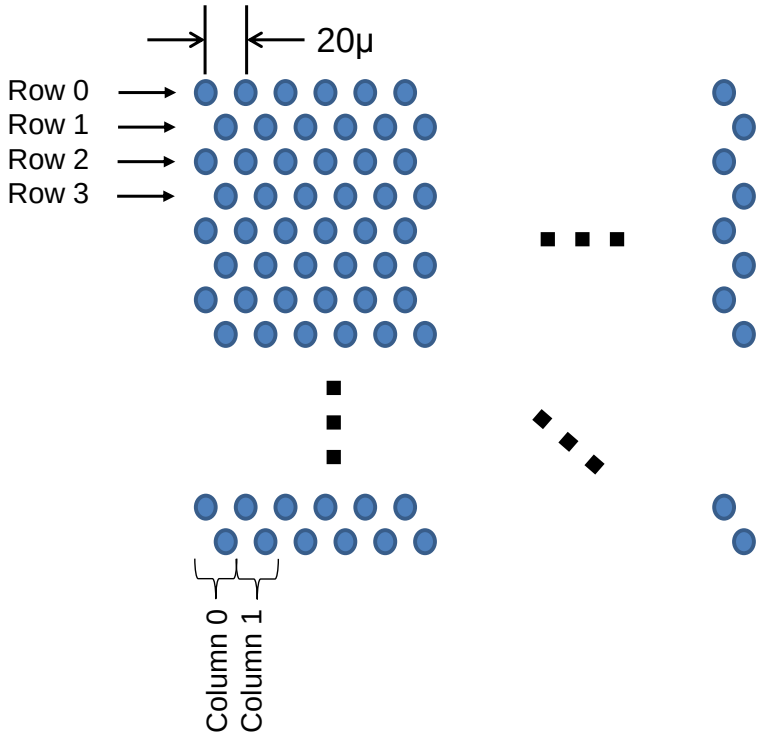


7.3.2 Hawksbill focal plane

The Hawksbill vision chip has a resolution of 136 x 136 pixels, arranged in a hexagonal format. The hexagonal array is formed by shifting right every other row of pixels by a half pixel, and then squashing the array vertically to form a proper hexagonal array. The pitch between pixels, which is defined as the distance between centers of a pixel and any of its six neighbors, is 20 microns.

The pixel at row 0, column 0 is at the upper left of the array, as shown in the layout of Section 3. Note that since odd-numbered rows are shifted right a half pixel, the row 1, column 0 pixel is in fact below and to the right of the row 0, column 0 pixel. Similarly the row 2, column 0 pixel is lined up to be vertically below the row 0, column 0 pixel and is also below and to the left of the row 1, column 0 pixel.

The Hawksbill chip supports array set addressing (ASA), an addressing scheme for hexagonal images that allows most image processing functions to be performed with a similar computational complexity as when performed with square arrays. In order to select ASA pixel (a,r,c), one would simply select row 2r+a and column c. A typical “raster scan” readout of the array, in which pixels are read out one row at a time, would therefore readout the array along dimension “c” first, “a” second, and “r” third.



7.4 Pixel Amplifier Circuit:

Note: The pixel amplifier should be considered an "advanced feature" that is used only when absolutely necessary, for example when operating the chip at a lower voltage. Using the pixel amplifier slows down the acquisition of pixels from the chips, and will limit the dynamic range. If you use the pixel amplifier, we recommend using the lowest gain possible.

The pixel amplifier circuit allows the raw selected pixel to be amplified by one of seven amounts, and then offset by a selectable amount. This pixel amplifier may be used to boost contrast sensitivity, but may reduce the overall range of light levels that may be simultaneously measured. The pixel amplifier circuit is configured using the CONFIG system register.

The figure below shows the overall block diagram of the pixel amplifier. The selected pixel is sent to a switched capacitor differential amplifier, where difference between the pixel and the VREF voltage is computed and amplified. Voltage VREF thus implements the offset.

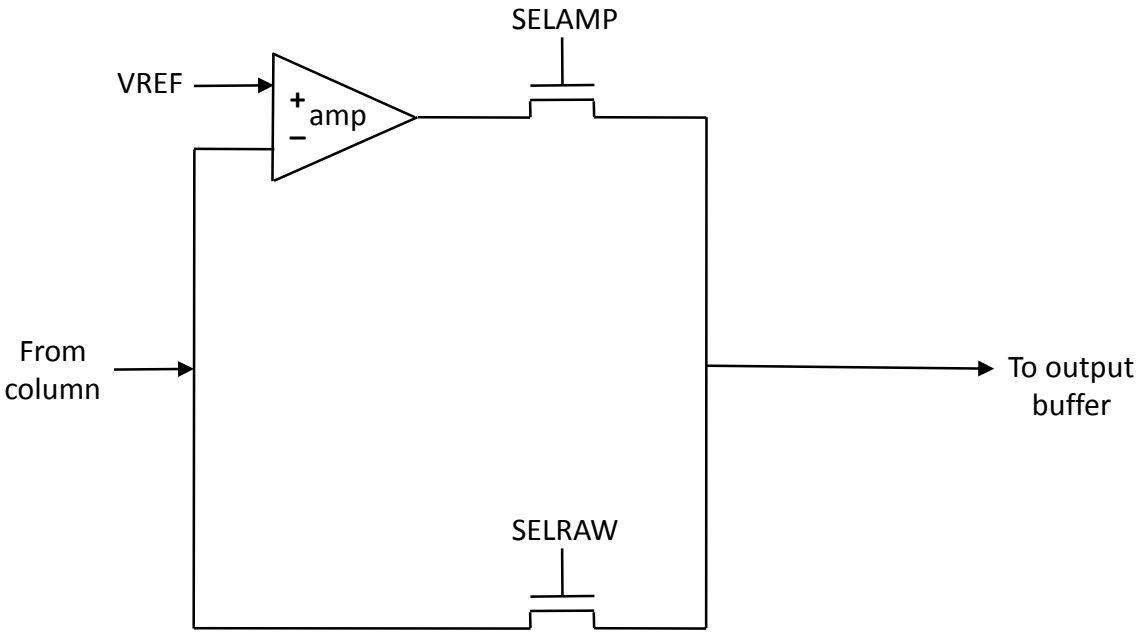
Signals SELAMP and SELRAW allow the amplifier to be bypassed. Bit 3 of the CONFIG system register determines these values- When bit 3 of CONFIG is low, SELRAW=1 and SELAMP=0, which causes the amplifier to be bypassed. When bit 3 of CONFIG is high, SELRAW=0 and SELAMP=1, which causes the amplifier to be utilized.

The gain of the amplifier is determined by bits 0 through 2 of the CONFIG system register, with bit 2 the most significant bit. The gain may be set to any value between 0 and 7. A gain of 0 results in just a DC voltage being output- a gain of 0 is thus meaningless- but it will not hurt the chip to use this gain. Other gains between 1 and 7 indicate the integer gain over what is obtained when the amplifier is bypassed.

The use of a gain of 1 thus does not provide any gain. However since VREF may still be changed, this allows the voltage level to be shifted. This feature is useful for when the image sensor chip is being operated at a lower voltage.

Recall that the CONFIG system register also controls the CVDDA signal which turns on the chip bias generators. If using the pixel amplifier, the value of CONFIG must be set to 8+16+gain or simply 24+gain. The “8” turns on SELRAW while the “16” turns on the bias generators.

In order to bypass the pixel amplifier, simply set CONFIG to 16.



The pixel amplifier may be operated as follows. For setting up, first ensure that bit 3 of CONFIG is high, and that bits 0 through 2 of CONFIG contain the desired gain. Then set the VREF voltage to the desired amount. Some experimentation of VREF and the gain may be needed.

Once the amplifier is set up, select the desired pixel using the ROWSEL and COLSEL system registers. Then raise digital input signal INPHI to a digital high. Hold INPHI high for a period of t_1, nominally one or several microseconds. Lower INPHI to a digital low. Wait for a period of t_2, nominally one or several microseconds. The chip output ANALOG will contain the resulting analog voltage.

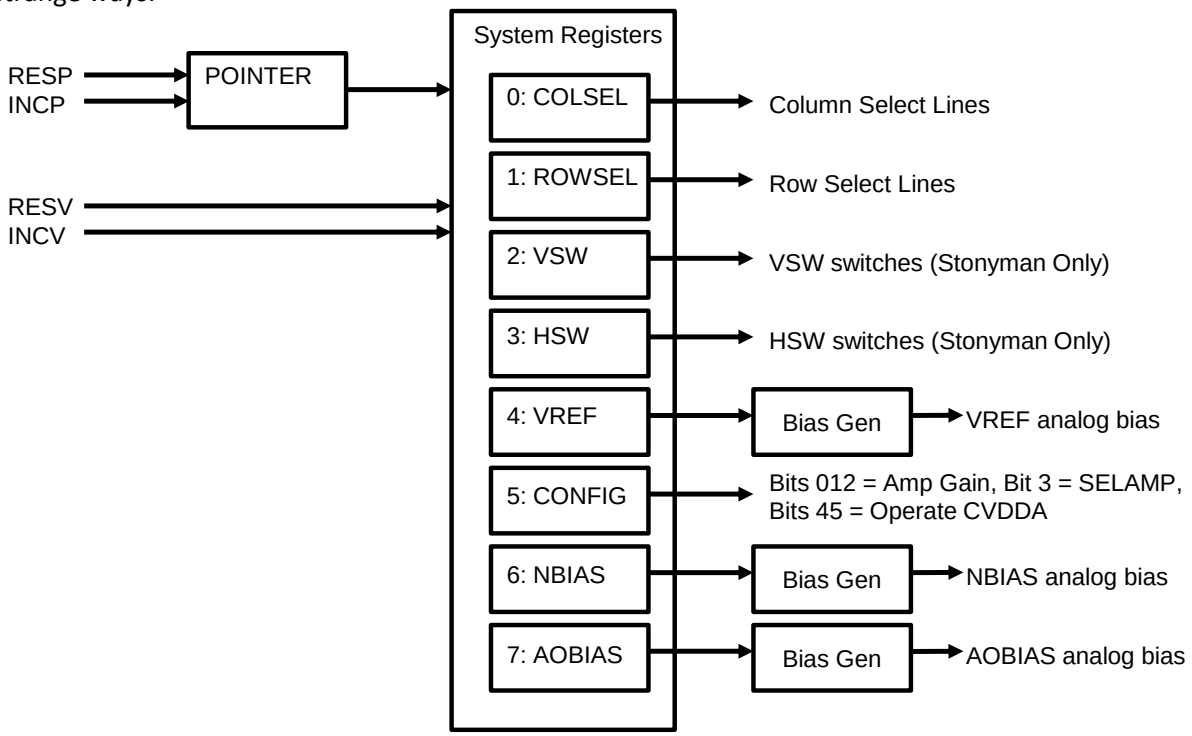
8. Operation of individual components:

This section discusses the operation of individual components and modules of the chip. The instructions and commands are presented below in the general order of operation for a sample application, and with increased functionality towards the end of this section.

8.1 Sending commands to the chip:

The four digital input signals RESP, INCP, RESV, and INCV are used to configure chip system registers. All configurations may be performed by pulsing these four signals in proper sequence. The figure below shows a behavioral model of these registers. The resting state for these four signals is digital low (GND). The active state is digital high (VDD). When we refer to one of these four signals as being "pulsed", we mean that the signal is pulsed high, e.g. raised from digital low to digital high, held high for a small duration (nominally a few hundred nanoseconds), and lowered back to digital low.

Two rules should be followed when pulsing these four signals. First, always return the signal back to digital low before doing anything else. Second, do not pulse more than one signal at a time. No more than one of these signals should be a digital high at any one time or the chip may behave in strange ways.



The four signals RESP, INCP, RESV, and INCV can be thought of as follows:

RESP = "Reset Pointer"
INCP = "Increment Pointer"
RESV = "Reset Value" (e.g. reset active register)
INCV = "Increment Value" (e.g. increment active register)

There are a total of eight system registers, numbered from 0 to 7. More commonly used system registers are given a lower number. The POINTER register determines which of the system registers are active at any one time. When signal RESP is pulsed, the POINTER register is reset to value "0". This

causes the COLSEL register (register 0) to be active. When signal INCP is pulsed, the POINTER register increments in value. This will activate a different system register. For example, to make the system register HSW (register 3) active, first pulse RESP and then pulse INCP three times.

When a system register is active, it may be similarly reset or incremented using the RESV and INCV signals. Pulsing the RESV signal will reset the active system register to zero. Pulsing the INCV signal will increment the active system register by one.

For example, if we want to set system register ROWSEL to 10, we could do the following steps:
Step 1: Pulse RESP (to reset POINTER)
Step 2: Pulse INCP once (to make ROWSEL active)
Step 3: Pulse RESV (to reset ROWSEL to zero)
Step 4: Pulse INCV 10 times (to set ROWSEL to 10)

Note that if a system register is not active, and then later made active, it is not necessary to reset that system register before setting it to a new value. It will hold its current value until you reset it or increment it. For example, suppose COLSEL is currently set to 20, and then the currently active system register is ROWSEL. If you want to increment COLSEL to 22, you only need to first pulse RESP to reset POINTER and make COLSEL active, and then pulse INCV twice to increase COLSEL from 20 to 22.

The POINTER register holds its value in a similar manner. For example, suppose POINTER=3 which makes HSW the active system register, and you want to adjust the VREF register (to set the VREF bias). You could first pulse INCP to increment POINTER and make VREF active, and then operate RESV and/or INCV as need to select the new value of VREF.

8.2 Turning on the chip:

The following steps are recommended to configure the chip on power-up:

Step 1: Configure the biases

Set NBIAS and AOBIAS to their respective values. For 5V operation we recommend experimenting with values between 55 and 60 for both of these.

Step 2: Set binning switches (Stonyman only)

If binning WILL NOT be used, then clear the VSW and HSW registers.

If binning WILL be used, then set VSW and HSW to their desired values.

Step 3: Configure

If the amplifier WILL NOT be used, then set CONFIG to the following value:

CONFIG = 16

This configures the pixel amplifier to be bypassed. The “16” connects the VDD and VDDA signals, which turn on the chip.

If the amplifier WILL be used, then set CONFIG to the following value:

CONFIG = Gain + 8 + 16

The value “Gain” must be between 1 and 7 and selects the gain of the amplifier. (It is possible to set the gain to “0” but the result is literally what one would expect…) The value “8” configures the pixel amplifier to be used (e.g. not bypassed). The “16” connects the VDD and VDDA signals, which turn on the chip.

8.3 Reading pixels from the chip

To read a pixel from the chip, generally you perform the following:

Step 1: Select row and column by setting ROWSEL and COLSEL registers to their proper values.

Note that you do not need to reset and set both of these registers every time you grab a pixel. For example, if you are currently reading out a row of pixels, and if the COLSEL system register is selected, then you only need pulse the INCV signal to advance to the next column pixel in the row.

Step 2: Operate Amplifier. (Skip this step if amplifier is not being used e.g is bypassed)

Set INPHI = high

Delay (empirically determined, about a microsecond)

Set INPHI = low

Step 3: Read pixel

First delay a small amount (empirically determined, start with a couple microseconds) and then digitize the analog signal at ANALOG.

8.4 Bias generators:

The bias generators are configured by setting the corresponding system register to the proper value between 0 and 63. The relationship between system register value and bias voltage is negative, with value 0 producing the highest bias potential and 63 the lowest bias potential. For operation at about 5V, the following bias settings are recommended:

NBIAS: Between 55 and 60

AOBIAS: Between 55 and 60

VREF: Between 40 and 55 (some trial and error needed). Note that VREF is not used if the amplifier is bypassed.

Note that the bias generators need to be “turned on” by setting bits 4 and 5 of the CONFIG system register respectively to 1 and 0. This may be perfomed by ensuring that when CONFIG is set, it is set to the value (16+8 +gain) if the amplifier is being used, or simply 16 if the amplifier is not being used.

8.5 Pixel binning to form superpixels (Stonyman chip only):

This section applies to the Stonyman chip only.

The above examples assume that a Stonyman chip is being read out at raw resolution. By setting the VSW and HSW system registers, it is possible to close the vsw and hsw switches in the focal plane pixel circuits according to a specified pattern. The VSW and HSW command each specify an 8-bit pattern that is repeated across the corresponding direction.

The HSW register is used to close "horizontal" switches. Setting HSW to binary pattern 00000001, for example, closes the hsw switches between columns 0 and 1, between columns 8 and 9, between columns 16 and 17, and so on. Binary pattern 10000000 closes the hsw switches between columns 7 and 8, 15 and 16, and so on. Note that the least significant bit of HSW closes the hsw switches between the left-most columns.

The VSW register is similarly used to close the "vertical" switches. Setting VSW to binary pattern 0000001 closes the vsw switches between rows 0 and 1, between rows 8 and 9, and so on. Binary pattern 00000010 closes the vsw switches between rows 1 and 2, between 9 and 10, and so on. Binary pattern 10000000 closes the vsw switches between rows 7 and 8, rows 15 and 16, and so on. Note that the least significant bit of VSW closes the vsw switches between the top-most rows.

To configure the focal plane to bin the pixels into 2x2 blocks, one would set the HSW and VSW system registers as follows:

Set HSW = 01010101
Set VSW = 01010101

The above system registers would only need to be set once. Each of these 2x2 blocks may be referred to as a "superpixel". To configure the focal plane to bin the pixels into 4x4 blocks, one would similarly set HSW and VSW as follows:

Set HSW = 01110111
Set VSW = 01110111

Finally, to configure the focal plane to bin the pixels into 8x8 blocks, one would need to set these system registers as follows:

Set HSW = 01111111
Set VSW = 01111111

Note that it is possible to bin the pixels by different amounts in each direction. Thus it is possible to bin the pixels into MxN blocks, where M and N may be 1, 2, 4, or 8. This allows the formation of rectangular superpixels, for example by enabling binning along one axis and not using binning along the other. It is also possible to short out entire rows or columns across the entire focal plane by setting the appropriate VSW or HSW system register to 11111111.

Once the pixels have been binned, operating the focal plane and reading out the pixels may be performed as above. The difference is that it is only necessary to digitize one pixel value from each block. For example, if VSW and HSW are set to bin the array into 8x8 blocks of superpixels, then only every eighth row and pixel needs to be read out. This can speed up the acquisition of a frame by 8x8=64.

Notes:
1) It should be obvious from prior examples, but we state it here: setting VSW and HSW to all zeros will turn off binning.
2) When binning, it is preferable to read out the same pixel from each block frame after frame. This is because the fixed pattern noise is partially dependent on which readout transistor (transistor M2) and which column bias transistor (transistor MC) is being used. Thus using the same pixel every time keeps this fixed pattern noise constant.
3) It is possible to use a different binning for each frame. Once the VSW and HSW arrays are changed a delay may be necessary to let the pixel circuits settle to the new values. This delay time will be shorter in brighter environments. At the current time we have not yet measured the delay necessary for each intensity level. In linear response mode, the act of resetting pixels is already part of the readout process so no new steps will need to be taken.
4) Binned pixel values may be amplified by the amplifier in the same manner as amplifying raw pixels.

8.6 Recommended bias and amplifier settings at various voltages

Raw (unamplified mode) at 5V:

We believe that the best results with these chips are obtained using the raw (unamplified) mode, with the chip powered at or near 5V. Lower voltages may be used, as low as 4V, but we recommend using 4.5V or more. In this mode, we recommend the following biases and configuration:

Set NBIAS to 55
Set AOBIAS to 55
Set CONFIG to 16

Amplified mode at 5V:

If you wish to use the amplifier at 5V, you will also need to set VREF. The optimal value of VREF depends on the light levels (as well as the bias values) and so you may need to experiment with VREF. We recommend the following settings to start:

Set NBIAS to 55
Set AOBIAS to 55
Set CONFIG to 24+gain (gain is between 1 and 7)
Set VREF to 40 (you will need to experiment)

Operation at < 4V, using amplified mode:

Operation at voltages less than 4V will require use of the amplifier. Again, you will need to experiment with VREF. At lower voltages we recommend keeping the gain of the amplifier small, around 1 or 2 at most:

Set NBIAS to 50
Set AOBIAS to 50
Set CONFIG to 24+gain (gain is between 1 and 7)
Set VREF to 30 (you will need to experiment)

8.7 Comments on Fixed Pattern Noise (FPN):

Fixed pattern noise (FPN) is inherent in all image sensor circuits. FPN refers to offsets between identically drawn pixel circuits that result from process variations. FPN manifests itself as a noise-like pattern that appears when an image sensor is provided with a uniform intensity. Ideally fixed pattern noise is measured once and then stored in memory. Then when a new frame is acquired, the stored FPN image is subtracted from the acquired image to produce a clean image.

Unfortunately, FPN is a function of pixel mode, amplification type (raw or amplified, including amount of gain used), operating voltage, and binning methods. A different FPN mask will need to be acquired for each of these permutations.

8.8 Read pixels row-wise vs. column-wise:

These chips allow pixels to be read out row-wise or column-wise. However for best results we strongly recommend reading out pixels row-wise. For example, you first select row 0, then read out all pixels in row 0, then select row 1, read out those pixels, and so on. If you do the opposite (e.g. select column 0, read out all rows, then select the next column) then transistor M3 (described in section 7.2) may inject switching noise onto the photodiodes and degrade the image quality.

9. Register List:

Below is a list of the eight registers and their meaning.

Register #	Name	Function
0	COLSEL	**Column Select** Stonyman: bits 6 to 0 select the column Hawksbill: bits 7 to 0 select the column
1	ROWSEL	**Row Select** Stonyman: bits 6 to 0 select the row Hawksbill (ASA format): bit 0 selects ASA value "a", bits 7 through 1 select ASA value "r" Hawksbill (2D format): bits 7 to 0 select the row
2	VSW	**Vertical switches (Stonyman Only)** Bits 0 to 7 determine respectively signals VSW0 to VSW7
3	HSW	**Horizontal switches (Stonyman Only)** Bits 0 to 7 determine respectively signals HSW0 to HSW7
4	VREF	**Reference voltage.** 0 = highest voltage, 63 = lowest.
5	CONFIG	**Configuration Register** NOTE: This is the only "complicated" register that supports multiple functions Bits 2 to 0 determine amplifier gain Bit 3 determines whether amplifier is used: 1 mean use amplifier, 0 means bypass Bits 5 and 4 operate CVDDA. To connect power signal VDD to VDDA, and thus turn the chip on, set bit 4 to 1 and bit 5 to 0 If the value of this register is changed, all three variables must be selected properly for the chip to operate.
6	NBIAS	**NBIAS voltage.** 0 = highest voltage, 63 = lowest.
7	AOBIAS	**AOBIAS voltage.** 0 = highest voltage, 63 = lowest.

10. Sample Instruction Sequences:

The instruction sequences below are written in pseudocode format. The reader should be familiar with the chip and it's operation, as described above, in particular Sections 8 and 9.

Note that for literal constants, we will use the following notation: A number by itself is a decimal (base 10) value. A number preceeded by "0x" is hexadecimal. A number preceeded by "b" is binary:

```
// The following represent decimal 20
20;         // decimal
0x14;       // hexadecimal
b00010100;  // binary
```

10.1 Updating the registers

We will use the following elementary instructions in the following examples:

Pulsing a digital line to the chip: The five digital lines to the chip normally rest at digital zero, e.g. at 0V. To pulse a line, set the line digital high, e.g. to VDD, and then immediately set to digital low. The signal needs to be held high for very little time, generally less than 100ns.

```
pulse(PIN_ID); // This function pulses one of the digital pins to the chip.
// for example:
pulse(RESP); // pulse the RESP line (to reset the pointer register)
pulse(INCP); // pulse the INCP line (to increment the pointer register)
```

Setting INPHI high and low: The INPHI line is also pulsed, but when used it is generally desirable to hold it high for a brief period. Note that INPHI is only used when the pixels are being read out in amplified mode. (Generally we recommend against using amplified mode unless you are operating the chip with a low power supply.) INPHI may thus be operated as follows:

```
set(INPHI,HIGH);         // set INPHI to digital high
delayMicroseconds(2);    // delay for two microseconds
set(INPHI,LOW);          // set INPHI to digital low
```

Setting a register to a value: The following function "setPtrVal" shows how to set one of the eight registers (Section 9) to a value. For example, in order to set the ROWSEL register to 10, we would call setPtrVal(ROWSEL,10) or equivalently setPtrVal(1,10).

```
function setPtrVal(register,value) do begin
      // reset pointer
      pulse(RESP);
      // now pulse INCP to set pointer to desired value
      for i = 1...register do loop
            pulse(INCP);
      end loop
      // at this point the desired register is active. So we first reset it.
      Pulse(RESV);
      // and now we pulse INCV to set the register to the desired value
      for i = 1...value do loop
            pulse(INCV);
      end loop
end function
```

Incrementing the active register: There are times where we will just want to increment the currently active register by a certain amount.

```
function incVal(increment_amount) do begin
      for i = 1...increment_amount
            pulse(INCV);
      end loop
end function
```

Activating a register without modifying it: There are times where we will just want to set the pointer to a certain value, to activate a register, without actually adjusting it.

```
function setPtr(desired_register) do begin
      pulse(RESP); // pulse RESP to reset pointer
      for i = 1...desired_register
            pulse(INCP); // increment pointer
      end loop
end function
```

Reading an analog value from the chip: Once the chip is turned on and the desired row and column is set, and the amplifier operated (if used), the pixel value at pin ANALOG may be read using an ADC. This may be an ADC located on a processor, or it may be an external ADC operated by the processor. We will assume the user understands the details of how to use the ADC, and here just use the following notation:

```
Invalue = operateADC(); // digitize potential ANALOG line and store
                        // in variable invalue.
```

10.2 Setting up the chip

Raw mode: (RECOMMENDED MODE) To set up the chip for use in raw mode, when powered at 5V, and with no binning (e.g. maximum resolution), send the following commands once. They can generally be sent in any order, but we recommend setting the biases first before setting the CONFIG register, since setting the CONFIG register to 16 turns on the chip's analog portion.

```
setPtrVal(NBIAS,55); // set NBIAS to 55
setPtrVal(AOBIAS,55); // set AOBIAS to 55
setPtrVal(CONFIG,16); // set CONFIG to 16 for raw mode
setPtrVal(VSW,0); // VSW=0 for no binning
setPtrVal(HSW,0); // HSW=0 for no binning
```

Amplified mode: We generally don't recommend use of amplified mode, but if you do want to use it, send the following commands. Note that you also need to set the VREF register as well. The example below assumes we are using a gain of 2.

```
setPtrVal(NBIAS,55); // set NBIAS to 55
setPtrVal(AOBIAS,55); // set AOBIAS to 55
setPtrVal(VREF,40); // set VREF to 40 (Determine experimentally)
setPtrVal(CONFIG,16+8+2); // 16 = chip on, 8 = "amplified" mode, gain = 2
setPtrVal(VSW,0); // VSW=0 for no binning
setPtrVal(HSW,0); // HSW=0 for no binning
```

Notes:
1) You do not need to set up VSW or HSW when using the Hawksbill chip, since the Hawksbill chip does not include binning. (If you do set these registers, there will be no effect.)
2) Generally the setup commands can be sent in any order, but it is recommended to set the biases first before turning the chip on with the CONFIG register. This is because the initial default bias on power up might be too high, and thus cause the chip to draw more current. This will generally not be a problem with one chip, but it could be a problem if many chips are being used together.
3) The above examples show the pointer being reset every time a different register is being sent. This is not necessary- you can, for example, set the pointer to VSW, then reset VSW, then pulse INCP once to set the pointer to HSW, then reset HSW, and so on.

10.3 Reading an image from the entire chip

The following example shows how to read an image from the entire chip. This example assumes we are using a Stonyman chip at 112x112 resolution. The following example also assumes the chip has been set up and turned on as described in Section 10.2.

Reading an image in raw mode: To read an image in raw mode, just loop through every row and column and digitize the pixel.

```
setPtrVal(ROWSEL,0); // select row 0
for r = 0...111 do loop
      setPtrVal(COLSEL,0); // select column 0
      for c = 0...111 do loop
            // delay to allow analog settling and acquisition by ADC
            delayMicroseconds(1); // generally a value between 1 and 2
            // get pixel and store in matrix array image
            image[r,c] = operateADC();
            // move to next column
            pulse(INCV); // pulse line INCV to move to next column
      end loop
      // And now we will increment the ROWSEL register
      setPtr(ROWSEL); // activate the ROWSEL register
      pulse(INCV); // increment the ROWSEL register to select the next row
end loop
```

Reading an image in amplified mode: Reading an image in amplified mode is similar, but requires operation of the INPHI line.

```
setPtrValROWSEL,0); // select row 0
for r = 0...111 do loop
      setPtrVal(COLSEL,0); // select column 0
      for c = 0...111 do loop
            set(INPHI,HIGH); // Turn INPHI high
            delayMicroseconds(2); // Recommend 2 microseconds
            set(INPHI,LOW); // turn INPHI low
            delayMicroseconds(1); // Final delay of 1 to 2 microseconds
            // get pixel and store in matrix array image
            image[r,c] = operateADC();
            // move to next column
            pulse(INCV); // pulse line INCV to move to next column
      end loop
      // And now we will increment the ROWSEL register
      setPtr(ROWSEL); // activate the ROWSEL register
      pulse(INCV); // increment the ROWSEL register to select the next row
end loop
```

10.4 Reading an image from a section of the chip

There are times when we just want to read out a section of the chip. The example below reads out the 10x10 block of pixels included in rows 20 through 29 and 30 through 39. We are only showing the example for raw (unamplified) mode.

```
setPtrVal(ROWSEL,20); // select row 20
for r = 0...9 do loop
      setPtrVal(COLSEL,20); // select column 20
      for c = 0...9 do loop
            // delay to allow analog settling and acquisition by ADC
            delayMicroseconds(1); // generally a value between 1 and 2
            // get pixel and store in matrix array image
            image[r,c] = operateADC();
            // move to next column
            pulse(INCV); // pulse line INCV to move to next column
      end loop
      // And now we will increment the ROWSEL register
      setPtr(ROWSEL); // activate the ROWSEL register
      pulse(INCV); // increment the ROWSEL register to select the next row
end loop
```

10.5 Binning and reading a binned e.g. downsampled image (STONYMAN ONLY)

This example shows how to use binning to construct blocks of "super pixels" which may then be read out. In this example, we will bin the array using 4x4 blocks, to bin the 112x112 Stonyman chip down to 23x23, and read out a 23x23 image

```
// First at program setup, we set VSW and HSW as follows
setPtrVal(VSW,b01110111); // See section 8.5 for binning values
setPtrVal(HSW,b01110111);

// Now, to read a frame, we just read every 4th row and column of pixels.
// Note that it is desirable to read in the middle of a block of pixels.
setPtrVal(ROWSEL,2);    // select row 1 (first block covers rows 0-3
                        // so we read row 1 (or 2)
for r = 0...22 do loop
      setPtrVal(COLSEL,1); // select column 1 (first block covers columns 0-3)
      for c = 0...22 do loop
            // delay to allow analog settling and acquisition by ADC
            delayMicroseconds(1); // generally a value between 1 and 2
            // get pixel and store in matrix array image
            image[r,c] = operateADC();
            // advance 4 pixels to next block
            incVal(4); // pulse line INCV 4 times to move to next block
      end loop
      // And now we will increment the ROWSEL register by 4
      setPtr(ROWSEL); // activate the ROWSEL register
      incVal(INCV,4); // increment the ROWSEL register by 4 rows
end loop
```

Notes:

1) The above example assumes that the binning is set to one value and left there. It is possible to select a different binning amount every frame. However if you change the binning amount, you will need to delay a bit to allow the pixels to settle to the new value. This delay amount depends on various factors, including the current and previous binning configuration as well as the illumination levels. The delay

could range from tens of microseconds to tens of milliseconds or more. You will need to experiment to determine the best delay for your application.

Erreta / Version Changes:

Version 0.0, August 10, 2011: Pre-release working document

Version 0.1, August 10, 2011: Pre-release working document
* Added Sections 3 and 8.1

Version 0.2, October 5, 2011: Pre-release working document
* Added much additional material

Version 0.3, November 4, 2011: Pre-release working document
* Clarified operation of pixel amplifier

Version 1.0, March 12, 2013: First regular release
* Section 3.3: Added details on wire bonding and encapsulation
* Section 5: Clarification on power supply quality and fixed typo in section title
* Added Section 8.8, reading out pixels row-wise vs. column-wise
* Added Section 9, pseudocode examples to operate the chip
* Clarified through that the pixel amplifier circuit is an advanced feature and not necessary for most users

Disclaimer: