Logarithmic CMOS Image Sensor on a Custom FPGA Baseboard

Michael J. Hague

The University of New South Wales - Canberra

Today’s star trackers play an integral role in the ability to accurately track and record the locations of spacecraft, satellites and space debris, from both ground based instruments, and more importantly, from the moving platforms themselves, where size, weight and power are very limited. This report aims to integrate a logarithmic CMOS image sensor onto a small form-factor FPGA baseboard, that is capable of capturing wide dynamic range images in real time. The methodology implemented included four stages of development. First, the system’s design logic was determined, followed by describing signals in HDL, running simulations to confirm functionality, and finally, integration of the sensor onto the baseboard. The final design confirms that sensor output data is being captured and stored on the FPGA, however video output was not possible due to an error in the baseboard wiring. Recommended future work includes developing an adapter to achieve HDMI output in order to confirm that the system is functioning as intended. In addition, data processing algorithms should be considered to enhance the current system’s functionality. Developing this small, low weight and power image sensor would provide a solution to the current gap in space image sensing capability.

Contents

I. Introduction
   A. Motivation  
   B. Project Aim  
II. Background
   A. Logarithmic Sensors  
   B. FPGAs & VHDL  
III. Methodology & Results
   A. Design Logic  
   B. VHDL & Simulations  
   C. Sensor Integration  
   D. Video Out  
IV. Future Work  
V. Conclusions  
VI. Personal Comments & Acknowledgements

References

Nomenclature

CDD = Charged Coupled Devices
CMOS = Complementary Metal-oxide Semiconductor
COTS = Commercial Off The Shelf
FPGA = Field-programmable Gate Array
FPN = Fixed Pattern Noise
GUI = Graphical User Interface
HDL = Hardware Description Language

1 SBLT, School of Engineering & Information Technology, ZEIT4500.
I. Introduction

A. Motivation

Space-based object detection and tracking plays a pivotal role in maintaining an updated trajectory dataset and detailed analysis of space objects [1]. Additionally, image sensing from the moving platforms themselves proves to be an increasingly difficult task with newer technologies giving rise to smaller satellites (known as CubeSats). CubeSats can take on the dimensions similar to a milk bottle and hence, traditional CMOS/CDD image sensors are impracticable due to the limited size, weight and power available. This project aims to integrate a logarithmic CMOS image sensor onto a small form-factor FPGA baseboard, providing a wide dynamic range, that is capable of capturing images in real time due to the clocking speeds of FPGAs. This report highlights the four stages of development as well as the significant results obtained at each stage. This method of demonstrating system functionality through staged outcomes, provides a grounds for verifying system functionality prior to advancing. Hence, the development of a small, low weight and power image sensor would provide a viable solution to the current technology gap.

B. Project Aim

The aim of this project is to integrate a logarithmic sensor onto a custom FPGA baseboard. More specifically, to develop the timing signals required by the sensor and to capture the pixel data on the FPGA for later processing. It is noted that data processing did not form part of the project’s scope, but rather the storing of data for later processing. The functionality of the design is defined as the successful implementation of the timing signals such that the pixel data can be seen at the output of the sensor (or alternatively, input to the FPGA). In addition, a visual representation of the pixel data should be viewed through video output in order to validate the system’s functionality. All components utilised throughout this project were COTS and supplied by UNSW Canberra.

II. Background

A. Logarithmic Sensors

In digital imaging, sensors convert image information into electrical data which is then further processed into an appropriate digital signal for viewing [2]. Traditionally, there are two types of devices that achieve this. They are CDD and CMOS image sensors. While these sensors provide good image quality, low noise and low complexity [2], they lack a wide dynamic range.

Dynamic range of an image sensor is the ratio between the minimum and maximum light that it can detect [3]. In other words, the limit of luminance range that a sensor can capture, or, the sensor’s capacity to retain scene information from very bright and darkened scenes [4]. As an example, linear CMOS pixels can see one to two decades of illumination [5]. However, logarithmic pixels can detect roughly five decades of illumination [2]. It therefore follows that logarithmic sensors are capable of much wider dynamic ranges when compared to their linear counterparts.

![Figure 1. Linear vs. Log Response Curve [6]](image)
Logarithmic sensor’s, as the name implies, captures image data on a logarithmic scale. That is, they produce a voltage level that is proportional to the logarithm of the amount of light striking a pixel. Hence, these sensors provide a signal which has a logarithmic response to the scene illumination [6]. Figure 1 (above) shows the response curve for a typical linear sensor vs. NIT’s proprietary logarithmic sensor. From this it is evident that the linear curve (dashed line) reaches saturation at an illuminance value of 10, whilst the logarithmic curve (solid line) never reaches saturation.

Traditional logarithmic sensors achieve a wider dynamic range through the absence of the capacitor in their pixel structure. As shown in Figure 2 (right), the linear pixel structure includes a capacitor at the output, which, over a certain light intensity, completely discharges and hence lowers output voltage. This results in a loss of detail in the brighter regions of a scene [2]. The logarithmic pixel structure however does not suffer from this effect and hence produces a more accurate depiction of the scene. The drawback from typical logarithmic sensors is the increase FPN, that is, the spatial variation in pixel output values under uniform illumination, or in other words, the non-uniformity in the response of individual pixels [7]. Fortunately, the specific sensor used in this project has internal FPN compensation buffers, which results in no noticeable FPN [8].

The sensor used in this project is the NIT NSC1005 logarithmic CMOS sensor, which is capable of delivering a dynamic range of more than 140dB [6]. This particular sensor’s pixel structure operates in NIT’s proprietary solar cell design, where the voltage generated by the photon is sensed out by a low noise readout circuit [6] (as opposed to traditional photodiodes, where electrons are generated by incident light). The full description of NIT’s solar cell structure is described in a previous communication [9], however the effects of this sensor can be seen visually in the example photos below (Figure 3). The left photo is a typical linear CMOS sensor and the right is NIT’s NSC1005 sensor. In both photos the light is on and it is clear that the left photo has saturated while the right has not.

**Figure 3. Typical Linear CMOS Sensor (left) vs. NIT NSC1005 Log Sensor (right) [6]**

**B. FPGAs & VHDL**

The main hardware components used throughout this project were a NIT NSC1005 Logarithmic CMOS sensor and a custom made baseboard. The baseboard included a Trenz TE0712-01 with an Artix-7 FPGA and a custom developed proxy board, which included the headers for the NIT sensor. Figure 4 below shows these components with a ruler for scale. Importantly, the left and middle photos are the reverse of each other. Other minor components were necessary including a Xilinx JTAG programmer, oscilloscope, wires etc. In addition, several supporting software tools were also utilised extensively, primarily Xilinx’s Vivado Design Suite and ModelSim.
An FPGA (as opposed to a microcontroller) was chosen for this project due to their ability to be completely reprogrammed to meet any design requirements (making them advantageous for prototyping) and their ability to handle extremely fast clocking speeds. As an example, clocking a 1280x720 pixel structure at a standard rate of 50MHz, takes only \( \sim 18.43 \) ms. In addition, FPGAs let the engineer have complete control over the design of the circuit, which is usually described in a form of HDL. Specific to this project, VHDL was the language chosen for describing signals. It is important to highlight that HDL requires a very in-depth level of detail in order to fully describe a system, which when compared to traditional computer programming, can often seem overwhelming. In HDL, the designer must specify everything about a signal, such as what exactly happens on a specific clock edge, on which pin that is occurring and using which I/O standard. Hence, an engineer that is new to HDL is likely prone to being distracted by the many rules, syntax and problems when writing HDL, when in actual fact, the statements that are written in HDL are actually specifying a type of hardware structure, such as flip-flops, gates, wires and pins.

The purpose of VHDL within the context of this project is to describe the behaviour of the signals that are required by the sensor. The behaviour of signals is described within processes. Processes sit within the architecture of VHDL designs, and are executed concurrently. The statements written within a process however, are sequential. This is a fundamental understanding that is required when writing HDL. In addition, component declaration, instantiation and global signals are described in the architecture, while inputs and outputs of the design are declared in the entity. Figure 5 (right) visually captures the design flow of VHDL.

III. Methodology & Results

This section aims at highlighting the various engineering steps that were required in order to complete the project. It highlights the four major incremental steps or milestones of the project, those being Design Logic, VHDL and Simulations, Sensor Integration and Video Out. Significant results are shown via diagrams, tables or figures during each stage in order to highlight the staged outcomes of the project.

A. Design Logic

Prior to writing HDL, it was necessary to consider how the system would behave in terms of which signals were required and how they would interact. As such, it became necessary to develop a state diagram to highlight these interactions and the mode of operation that the sensor would exhibit. The state diagram (shown in Figure 6 below) was developed sequentially and was primarily used to describe the basic operation of the sensor. The sensor operation was described in the NSC1005 datasheet [8], specifically, though the timing diagrams. The advantage of a state diagram is that it allows for easier implementation of HDL through decomposing the problem into four states of operation. In addition, this visual representation allows for future users to more readily understand the mode of operation.

As shown in the top half of Figure 6, the four states of operation are line begin, readout, counter and retrace. These four states described the sensor’s line readout (or horizontal pixel capture) and is dependent on the Hsync signal being active high. In other words, the 1280-line pixels are clocked out only when Hsync is high, which occurs after the initial buffer period and stays high until all pixels are clocked before retracing and beginning a new line. It should be noted that this mode of operation occurs indefinitely.

Figure 4. Trenz TE0712-01 Baseboard (left), Dr. A. Lambert Baseboard (middle), NIT NSC1005 Log Sensor (right)

Figure 5. Visualisation of VHDL Design Flow
As previously mentioned, the state diagram was developed sequentially, that is, only one signal was introduced at a time, described in HDL and simulated, prior to moving on. Hence, at this stage of development, the top half of Figure 6 was described in HDL and simulated to confirm functionality. Importantly, the act of describing signals in HDL and simulating them occurs in the second stage of the project’s lifecycle (i.e. HDL & Simulations). It therefore follows that whilst the project followed a sequential design and implementation flow, stages one and two were interchangeable in order to allow for troubleshooting and debugging at the signal level as opposed to the system level. This method of engineering design allowed for easier identification and rectification of incorrect signal behaviour throughout the project.

The next signal for consideration was \( V_{clk} \), which describes the sensor fetching a new line for readout. The bottom right of the state diagram shows the operation of this signal. A rising edge detector was utilised such that \( V_{clk} \) was synchronous to the rising edge of \( H_{sync} \). Finally, in order to limit the number of line readouts, the signal \( V_{sync} \) was introduced. This signal’s behaviour is shown in the bottom left of the state diagram, where it simply counts the number of times \( V_{clk} \) is high. When a specified limit is reached (in this case 720), \( V_{sync} \) goes low, signalling the end of a frame.

![Figure 6. State Diagram for Core Sensor Signals](image)

**B. VHDL & Simulations**

Once the project’s design logic was determined, signals were described using VHDL and further simulated to confirm correct behaviour. Both signal description and simulations were conducted using ModelSim. In addition, a test bench was constructed in order to provide the design with a repeatable stimulus to begin its readout and to test the reset signal. This file was also created in ModelSim using VHDL to describe the behaviour.

It is noted that several signals that are required by the sensor are not mentioned in the above state diagram. This is because they are not reliant on any synchronous edge, and hence could be added to the design once the core signals were described and simulated. These signals included \( RD1, RD2, RST\_Line \), which refer to the buffer signals of the sensor. The same logic applies for the amplifier gain signals \( G0 \) and \( G1 \), which were set to zero.
Throughout this stage of the project extensive use of manuals and tutorials [10] [11] [12] were utilised in order to assist with the describing of signals. Important concepts were required to be learnt (as previously mentioned) such as concurrent vs. sequential statements, correct use of signal assignment and declaration and methods of specifying signal length (such as counters).

The full timing simulation can be seen in Figure 7 below. It is noted that the full frame capture (1280x720) did not produce a clean picture for presentation and hence only 1280x7 pixels are represented. Despite the lower number of readouts, it is still sufficient to visually confirm that the signals are behaving as per the sensor datasheet’s timing diagram. The column and row counters are also shown in the figure to highlight the number of line readouts (mentioned previously as 7) and the number of pixels in a line (1280). It is noted that the column counter reads only 1278 pixels, however this is by design as the \textit{Hsync} signal is required to go low prior to the end of the line, thus giving it two clock cycles to do so.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure7.png}
\caption{Simulated Timing Diagram of Sensor Signals using ModelSim}
\end{figure}

C. Sensor Integration

This stage of the project incurred by far the most troubleshooting, which included both known unknows and unknown unknowns. Basic tutorials of Vivado using a Digilent Nexys development board were first conducted in order to gain an understanding of the software and what was required. Once this was completed, the initial steps of integration could occur.

The initial steps included mapping the required signals through the baseboards, to the correct FPGA pins and adding the system clock. Analysis of several schematics and datasheets [13] [14] [15] was required in order to determine which headers corresponded to the required signals and in turn, to which FPGA pins. With this knowledge, a constraint file was created. A constraint file is necessary for programming a design onto an FPGA as it assigns which physical pins on the FPGA are being used with reference to the signals described in the design. As previously mentioned, it is necessary to describe everything about how a particular signal behaves. This includes how the signals get into, and out of, the design, hence the purpose of the constraint file.

The system clock was added using Vivado’s library of IP cores, specifically the MMCM, which allows for rapidly adding various system clocks as required by the design. In this case the system clock was set to 50MHz, which was selected based off the clock rate of the pixel clock (i.e. \textit{Hclk}), described in the sensor datasheet. The Trenz Electronic TE0712 TRM [16] was consulted in order to determine the correct clocking parameters and constraints, such as FPGA bank/pin, I/O standard and frequency.

The I/O standard used for most signals was LVCMOS33, whilst the system clock used DIFF\_SSTL15, as specified by the datasheets and video out required TMDS33. The importance of constraining signals with I/O Standards is to match the impedance of the input/output lines and the device in order to avoid transmission line reflection [17].

Once completed the design was synthesised, implemented and bit-streamed in Vivado. Synthesis turns the HDL files into transistor level logic based on the design timings and I/O constraints. Implementation optimises the design to fit onto the FPGA, places the design and routes the signals. Bit-streaming generates the final outputs needed for programming the FPGA. After troubleshooting, debugging and finally programming the design onto the FPGA, signals were first confirmed as to whether or not they appeared at the correct baseboard headers. This was achieved using oscilloscope probes. Whilst selected signals could be seen on the oscilloscope, this method of
validation was very limited for two reasons. First, the pin headers were very small, making it difficult to accurately and consistently probe the correct header. Secondly, and more importantly, a maximum of two signals could be probed at any one time, making the ability to confirm correct signal interaction and behaviour impossible.

The solution to overcome this limitation was to use Vivado’s ILA, which allows the designer to select up to 64 signals to observe in a waveform GUI. It is important to note that this IP core is not a simulation, but rather a visual representation of the real internal signals of the FPGA. The only limitation with this method is that that maximum frame width is 131,072 cycles, or ~2.62ms. Comparing this to a full frame readout of 921,600 cycles, or ~18.43ms, it is obvious that a full frame cannot be view (in fact only approximately 14.2%). Despite this, the ILA provided good grounds in which to observe signal interaction and behaviour.

With this in mind, all relevant signals were added to the ILA and observed in the GUI. Figure 8 below shows the real signals in the waveform GUI (top) against the sensor’s datasheet timings (bottom). It is noted that Hclk is absent from the waveform GUI as it represents the system clock. In order to confirm correct signal length, measurements were taken using the GUI timescale and matched against the datasheet timings.

![Figure 8. Vivado’s ILA GUI Showing Sensor Signals (top), NSC1005 Datasheet Timings (bottom) [9]](image)

Table 1 below highlights the measured signal widths, internal to the FPGA, against the required widths as dictated by the datasheet. Measurements were taken using the GUI timescale, specifically using two line cursors to measure the width of a signal. The difference in timescale gave the number of clock cycles. With the knowledge that the system clock was 50MHz, then the period of one cycle was 20ns. Hence signal widths could be calculated. As an example, the measured width of signal RD1 was 100 clock cycles. Multiplying this value by the clock period gives 2us.

**Table 1. Measured Signals Widths vs. Datasheet Signal Widths**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Measured Width (FPGA)</th>
<th>Required Width (Datasheet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD1</td>
<td>2us</td>
<td>2us (min)</td>
</tr>
<tr>
<td>RD2</td>
<td>2us</td>
<td>2us (min)</td>
</tr>
<tr>
<td>RST</td>
<td>2.06us</td>
<td>2.05us (min)</td>
</tr>
<tr>
<td>Vsync</td>
<td>14.4us</td>
<td>727 \times Vclk (max)</td>
</tr>
<tr>
<td>Vclk</td>
<td>20ns</td>
<td>Hclk</td>
</tr>
<tr>
<td>Hsync</td>
<td>25.6us</td>
<td>1287 \times Hclk (max)</td>
</tr>
<tr>
<td>Hclk</td>
<td>20ns</td>
<td>20ns (min)</td>
</tr>
</tbody>
</table>
There were several other smaller intricacies that were required such as the timing between the end of one frame and the start of the next, waiting periods between certain signals etc. These restrictions were specified as having a typical or minimum length and as such were generally given a timing greater than this to allow for some wriggle room. These restrictions are not captured in the above table as there are too many to represent in a logical and visually aesthetically pleasing way.

The final step included adding the sensor’s 14-bit data output to the design, and storing it in registers on the FPGA. This allowed the output data to be added to the ILA and viewed on the waveform GUI. The sensor was then placed onto the baseboard and Figure 9 below shows both the timing signals and the sensor’s 14-bit output. This figure confirms that image capture is occurring.

![Figure 9. Sensor 14-bit Data Output view in Vivado’s ILA GUI](image)

**D. Video Out**

Whilst the figure above does confirm that image capture is occurring, and data being sent and stored in the FPGA cannot be confirmed for quality or correctness, Hence it was necessary to provide a means for displaying the data on a video out feed. Several options were viable, however HDMI out was selected due to existing support software and design schematics.

The HDMI design is shown in Figure 10 and includes a 5x differential clock to drive the TMDS outputs. Various other signals are present and all were required to be declared and instantiated within the top level VHDL file. The differential clocks were added using the previously mentioned MMCM in Vivado and the speeds were determined based on the desired resolution and frame rate. That is, a 720p resolution at 30Hz, which requires a 720Mbit/s data rate, resulting in a 240MHz clock (for uncompressed 8-bit RGB).
Figure 10. HDMI Altium Design used in this Project

Once all modifications were made, the design was once again synthesised, implemented, bit-streamed and programmed onto the FPGA. However, while the correct signals and data outputs were observed (identical to Figure 9 previously), no video was displayed on the monitor. It was discovered after troubleshooting that the baseboard HDMI pin header was of type D, while the baseboard schematic wiring was of type C. Hence there was signal misalignment and getting any sort of video out was impossible. Figure 11 below shows this visually, where, for example, pin 1 on the schematic is given the assignment **TMDS Data2 Shield**, while pin 1 on the baseboard header is given the assignment **Hot Plug Detect**, hence a signal misalignment.

Figure 11. Comparison of Schematic (left) vs. Baseboard (right) Signal Assignment for HDMI

Therefore, despite best efforts, video out was not possible due to an error in the baseboard wiring, resulting from a misinformed schematic diagram, and hence, validation of the system’s functionality cannot be fully confirmed.

IV. Future Work

There are three main areas that have been identified as potential future work. The first recommendation is to correct the HDMI header/pin assignment such that video out becomes possible. This provides the benefit of allowing the sensor’s digital data to be viewed visually as a video feed as opposed to binary, and in turn, provide feedback on whether or not signal behaviour is truly correct or whether tweaks are necessary. It is recommended
that either an adapter board be developed to address the signal misalignment, or a micro HDMI cable cut open and remapped in accordance with the schematic.

The second recommendation is to develop data processing algorithms in order to provide additional functionality. Such algorithms may include advanced data storage and exporting, filtering and/or object detection. This recommendation would form the bulk of the potential future work.

Finally, it is recommended that improvements be made to the existing design, both how it has been described as well as paying closer attention to the intricacies of the signal behaviour (such as rise/fall times etc.). The degree of work required in this recommendation is largely dependent on the results of the video output feed. That is, whether or not a quality video out is achieved or if signal tweaking is required.

V. Conclusions

The aim of this project was to integrate a logarithmic sensor onto a custom FPGA baseboard, that is capable of real time image capture. To do this, the project was broken into four stages. The initial two stages was focussed around the design logic and describing signals in VHDL, utilising simulations to validate correct signal behaviour. Once this was successfully demonstrated, steps were taken to physically integrate the sensor onto the baseboard. The internal FPGA signals as well as the sensor’s 14-bit output data was successfully viewed on Vivado’s waveform GUI, thus confirming design functionality. However the final stage (video out) was not able to be completed due to a mismatch of HDMI signal mapping between the schematic and the baseboard header. It is recommended that a adapter board be developed in order to correct the mismatch of HDMI mapping and hence achieve video out. This would allow for absolute clarification on whether or not the system is working as intended or whether some tweaking is necessary. Developing a small, portable, low weight and power image sensor, utilising an FPGA and logarithmic sensor, provides an optimal solution for CubeSat (or similar) mounted image sensors.

Personal Comments & Acknowledgements

I have personally found this project to be very challenging. I say this because I had very limited background knowledge and experience with VHDL and imaging in general (having only a portion of a semester’s worth of experience). It was therefore difficult to begin writing effective HDL statements without heavy consultation of manuals and tutorials. In hindsight, describing the signals in a software designer, such as Altium, and then converting the design to HDL may have been an easier approach. I would recommend this approach for any new designers wishing to use VHDL with limited experience.

I would also like to personally thank my supervisor, Dr. Andrew Lambert, for his guidance, advise and challenging me to think. My co-supervisor, Dr. Manuel Polo, for providing helpful suggestions throughout the project, and Fanpeng Kong, for assisting me with the Vivado software and troubleshooting complex VHDL syntax errors. Your support throughout the year has been very much appreciated and I thank you for pointing me in the right direction in order to solve problems and not spoon feeding me answers.

References


