Investigation & Integration of High Speed Optical & Processing Hardware for Implementation of Stream Processing Centroid Estimation in High Speed Star Tracker

Luke Armstrong*

University of New South Wales at the Australian Defence Force Academy

This project explores several hardware components and their potential for incorporation into a high-speed star tracker. With a focus on the CMV12000 CMOS image sensor from CMOSIS and the SmartFusion2 SoC from Microsemi. This in tandem with integration of a stream based centroid processing algorithm designed for use on an FPGA and detailed in the 2017 research paper 'Centroid estimation for a Shack-Hartmann wavefront sensor based on stream processing'.

Contents

1. Nomenclature 1

I. Introduction 2

II. Background 2

III. Project Approach 4

A. Hardware 4

1. Initial Testing & Integration 4

2. Adapting the streaming centre of gravity algorithm to the SmartFusion2 5

3. Establish control of CMV12000 CMOS Image Sensor 5

4. CMV Sensor Image Retrieval 6

B. Simulation 9

1. Centroid Identification using Matlab Approximation of SCoG Algorithm 9

2. Star Catalogue 9

IV. Conclusions 10

V. Future Work 10

VI. ACKNOWLEDGEMENT 11

References 12

Appendices 14

Nomenclature

$M_v$ visual magnitude. 9

$Mbps$ Megabits per second. 2, 3, 6, 7

$fps$ frames per second. 2

*ASLT, School of Engineering & Information Technology. ZEIT4500/4501
I Introduction

Star trackers are becoming common on satellites as a high accuracy attitude detection system[1]. There are many commercial options that are designed around large satellite deployment[2][3]. These typically large systems, provide updates in the range of 1 to 10 Hz[2][3] which requires the satellites to incorporate other systems for detecting attitude motion between star tracker updates. With the costs involved in getting large satellites into orbit micro-satellites have begun to present themselves as a less costly alternative[4][5]. Micro-satellites, are a disruptive technology in satellite sphere[4]. They are faster to produce, offer significant cost savings in development and launch, and can perform many of the tasks of traditional satellites[4].

Micro-satellites require very efficient systems in terms of power usage, size, weight and utility[5]. A modern star tracker that is cognisant of these constraints, while improving utility over existing systems, such as increasing the speed of the star tracker, would be a very beneficial technology. By increasing the speed of star tracker attitude updates, there is potential for the star tracker to reduce or replace several traditional attitude detection systems that occupy the limited resources of micro-satellites. Alternatively by having a star tracker system that has alternative functions the utility of the system is increased.

This project explores several hardware components and their potential for incorporation into a high-speed star tracker. The main project focus is the CMV12000 image sensor (CMV) complementary metal oxide semiconductor (CMOS) image sensor from CMOSIS[6] and the SmartFusion2 (SF2) system on a chip (SoC) from Microsemi[7]. Integration of a stream based centroid processing algorithm, detailed in (Kong, Polo, Lambert, 2017)[8], and designed for use on an Field programmable gate array (FPGA) is another important project component.

When integrated into a star tracker system, it is intended that these components will produce attitude solutions at a higher rate than commercial systems with reduced size, weight and power requirements. The components, particularly the high resolution sensor, also offer versatility in use case allowing re-purposing in-situ.

This project covers early testing and integration steps in development of this high-speed star tracker system. Integration of the core components and establishment of command and control signals between these components.

II Background

CMV12000. The CMV sensor, developed by CMOSIS, is a high-speed monochrome CMOS image sensor with 12MP resolution at 8, 10 or 12 bit pixel depth, offering high-speed and quality image data for use in the system[9]. The benefits of the sensor in this project application include: high speed operation at up to 165/ps utilising 32 low voltage differential signaling (LVDS) outputs at 300Mbps; pipelined global shutter pixels which allow exposure during readout, which attributes to the high speed; True correlated double sampling (CDS) which leads to significantly reduced pattern noise and dark noise; serial peripheral interface (SPI) for ease of programming by the SF2; Windowing ability allowing a specific region of interest (ROI) to be output, which can further increase speed or alternatively decrease power consumption[9][6].

CDS is a method of removing undesired offsets that may be experienced when measuring a voltage or current. This is achieved by comparing a known reading and an unknown reading and measuring the difference. The known reading is a default value and the unknown reading is the measurement of interest. In the case of a CMOS sensor pixel the known reading is the reset voltage of the pixel and the unknown is the voltage induced due to exposure to photons. This process can compensate for noise such as the readout noise which is induced due to circuit architecture and the intrinsic noise of pixel devices, it can also compensate for temporal and MOSFET noise which impacts CMOS sensors[10].

The ability to define specific ROI is particularly useful in a star tracker (ST) because the majority of the pixels contain no useful information. It is beneficial to locate the stars of interest within the field of view (FOV) and then observe only these stars and their immediate area on subsequent frames to track attitude drift. The CMV12000 windowing functionality enables this by allowing selection of specific sets of pixel rows to be read out. Up to 32 individual windows of consecutive pixel rows can be output, with the desired regions being selected by programming the applicable CMV registers. This is done using the SPI, and can be adjusted every single frame to allow windows to be adjusted to track the appropriate ROI. To further define a ROI, only the relevant portions of the output rows need to be processed with the remainder of the row discarded allowing a ROI of any size to be utilised. The benefit of not reading out an entire frame is either, a further increase of speed for both the sensor and the overall system, or the ability to reduce the power consumption of the sensor[9].

SmartFusion2. The SF2, developed by Microsemi, is a SoC device which incorporates both FPGA and micro-controller sub-system (MSS) benefits into a single device[7]. Like the CMV, it too has several features that are
beneficial for a ST operating in space such as: Reliability in the form of single event upset (SEU) immunity due to zero failure in time (FIT) FPGA configuration cells, single error correct double error detect (SEUED) protection on various buffers and memory controllers and also SEU resistant latches[11].

The SoC functionality will allow the implementation of the stream centre of gravity (SCoG) onto the FPGA and allow the secondary star identification on the central processing unit (CPU). The SCoG algorithm requires an FPGA for implementation, this is due to the parallel nature of the algorithm[8]. The mathematical operations of conventional star matching algorithms though, are better suited to a conventional CPU, thus the SoC is a single alternative for both requirements.

Factors including, but not limited to, cosmic radiation and inability to perform system repairs once in orbit mean that reliability and robustness to error causing external elements is a critical feature of all components. In orbit a spacecraft will experience impacts from electrons and protons from sources such as the Van Allen Belt and solar flares[12], as well as high energy neutrons and alpha particles[13]. These particles have the potential to not only damage sensitive equipment, but also alter stored information which can corrupt the system programming or stored data. The SF2 incorporates several reliability features that minimise the likelihood of such issues occurring including Zero FIT rate FPGA configuration, SEU protected memories, memory controllers with SEUED and built-in self test functionality. These are manufacturer terminology for various architectural design features which have been tested in accordance with standards such as the JESD85 (Methods for calculating failure rates in units of FITs)[14].

Magneto-resistive RAM. Magnetoresistive random-access memory (MRAM) is a form of random-access memory (RAM) that utilises magnetic fields to store bits instead of electric fields. There are several benefits to utilising MRAM for a space based system, including the fact that its non-volatility is guaranteed for > 20 years and doesn’t require power, it has unlimited read/write endurance and also that the data isn’t susceptible to degradation due to the charged particles that will bombard it in space[15]. Compared to the potential data rates of the CMV (100 to 600Mbps) the MRAM are relatively slow at 35ns read/write (∼ 28MHz) although is relatively fast when compared to the MSS clock so it provides an excellent storage solution for the MSS.

Advanced High-performance Bus. Advanced high-performance bus (AHB) is an open standard, on chip interconnection specification that is utilised in SoC devices like the SF2. It is a high performance system bus that supports multiple bus masters and high-bandwidth operation[16]. Native implementation in the SF2 makes this a convenient bus solution for utilisation in various tasks. By utilising native solutions, including the AHB and slave blocks similar to the ‘CoreMemCntl’, it is possible to mitigate the risk of introduction of errors from bespoke solutions. For example, it is possible to implement a memory access solution quickly requiring far less VHDL hardware description language (VHDL) than an equivalent bespoke solution. There are of course limitations that are imposed by utilising a bus, including that buses must handle congestion and that it is clocked at the MSS clock rate, which impose speed restrictions on high-speed tasks[16].

Serial Peripheral Interface. SPI is a synchronous serial communication interface for short distance full duplex communication between a master & a number of slaves[17]. It utilises a clock, master-out slave-in (MOSI) and master-in slave-out (MISO) plus one slave-select (SS) line per slave. Communication is initiated by the master by activating the applicable SS line and applying the applicable serial data out on the MOSI line along with the clock signal, the slave may then respond on the MISO line. SPI is a common across most different micro-controller devices from Arduino through to SoC devices like the SF2[18][19]. This makes it an ideal method for ensuring compatibility between devices such as the SF2 and CMV, allowing programming of such devices.

Low Voltage Differential Signalling. LVDS is a method of signalling that has benefits over single ended signalling. In LVDS the transmitted information is transmitted as a difference between the voltages on a pair of wires or tracks. It is generally terminated by a 100Ω load with the voltage polarity referring to either 1 or 0[9]. This method of transmission is beneficial especially in cases where the LVDS lines are tightly coupled as the fields produced by the current in the differential lines is equal and opposite cancelling out the magnetic fields otherwise created and minimising electromagnetic noise on other components[20]. It allows high-speed, low noise and low electromagnetic interference communication which is ideal for satellite based systems[20]. A method of increasing throughput over LVDS is the use of double data rate (DDR). In DDR data is output on both edges of the clock as opposed to single data rate which only utilises one clock edge.
III Project Approach

Two different approaches were utilised in this project, a hardware based and software simulation approach. The hardware based approach, which is the aim of the project, is shown in figure 1. The simulation parallels the hardware approach and allows a proof of concept to be developed. It improves understanding of the complete system operation and has potential to accelerate latter project components since some of the algorithms have been developed.

A Hardware

The hardware approach consisted of four main phases that work towards the higher level aim of a high-speed star tracker. The phases consisted of integrity testing of hardware prior to assembly, incorporation of the SCoG algorithm VHDL into the SF2 project, establishing communication and control of the CMV sensor, and the design and implementation of an image capture method in an attempt to retrieve image data from the CMV sensor for external processing.

1 Initial Testing & Integration

Ensuring hardware integrity prior to assembly is an important step in the engineering process[21]. The CMV12000 test board (CMVTB) was an untested hardware component, and required integrity testing prior to installation of the CMV sensor, to mitigate the potential for catastrophic damage to the sensor. Two key checks were carried out on the board, DC power supply health tests and signal routing checks.

Power supply testing was carried out to ensure CMV sockets on the CMVTB as well as all of the test points on both the CMVTB and SF2 flight computer (SF2FC) were correct in accordance with (IAW) the CMVTB and SF2FC schematics[22][23]. The CMV and SF2 voltage requirements are shown in tables 1 and 2 respectively. The DC voltages were measured over a one minute period using a digital multimeter, while simultaneously monitoring the DC supply with an oscilloscope for transient instabilities outside the allowed range. The DC voltage testing concluded that the DC supplies operated within the allowed range throughout the test period.

The signal routing checks were carried out to ensure correct routing of PCB tracks and connectors. This was done by outputting a square wave (by blinking a general purpose input/output (I/O) (GPIO)) from the SF2 and monitoring the CMV sockets on the CMVTB. This testing was carried out on the connections between the CMV and SF2, comparing results to those expected IAW schematics[23][22]. This phase of testing, once passed, allowed connection of the CMV to the CMVTB and led to the next project phase.
2 Adapting the streaming centre of gravity algorithm to the SmartFusion2

The SCoG algorithm, supplied for the project, was initially designed for use on Xilinx Spartan-3A DSP FPGA (XC3SD3400A-4CS484C) [8] which had several inconsistencies when implemented directly onto the Microsemi FPGA. Importantly the method of first-in first-out (FIFO) implementation between FPGA manufacturers was inconsistent. The FIFO are an integral part of the SCoG design, seen as line buffers in figure 6[8].

The SCoG algorithm required modification in order to instantiate appropriate FIFO. CoreFIFO is the Microsemi Libero native FIFO solution and needed to be generated within the Libero project for use within the SCoG block. The CoreFIFO is highly modifiable, this allows it to be utilised throughout the project for testing. It was therefore a logical choice to utilise the CoreFIFO generation tool to generate all the required VHDL and adapt this to the SCoG instead of developing bespoke FIFO VHDL.

Testing of the SCoG algorithm has been carried out by ensuring that the SCoG VHDL compiles and the project builds. Further testing will be required once there is valid CMV sensor data to supply to the algorithm. The current configuration for the algorithm can only make use of a single pixel supplied per clock cycle thus it can't utilise more than four LVDS channels simultaneously. In order to utilise four LVDS channels, staggering of the pixels so that one arrived each clock cycle would be required, this is due to it taking four clock cycles for 8 bits to transmit. Therefore further work and testing is required to find a solution that will make full use of the CMV outputs.

3 Establish control of CMV12000 CMOS Image Sensor

SPI. SPI is built into the CMV and is the required method for programming of the CMV sensor[9]. Programming is done by interacting with the on-board registers and writing them to the appropriate values (table 4 shows an example of the required register writes for 8 bit mode[9]). The CMV utilises registers for programming of the various features and functions of the sensor such as gain, windowing, frame bursts and control of the LVDS outputs. In order to get this system operating there are several key requirements. These include correct routing of signals between the MSS and CMV via the FPGA fabric (fig 1), interpretation of CMV data-sheet into required register writes for different operating modes (see table 4) and writing software to run on the advanced RISC machine (ARM) core program in the MSS that can perform the register writes.

The SPI controller inside the SF2 allows two separate SPI implementations. One is connected to external pins on the SF2 and to a FLASH memory module while the other can be connected through the fabric. The SPI routed to FLASH memory allows reprogramming of the SF2[24] while the other SPI remains to control the CMV registers. The CMV utilises mode 0 SPI, this means the clock idles low and transfers occur on the leading edge of the clock pulse. Unlike standardised SPI devices though, the CMV requires the SS signal to be active high[9]. In order to utilise the inbuilt SPI controller of the SF2 the solution was inversion of the SS line to enable SPI communications between the CMV and SF2. The process of programming the CMV over SPI is carried out by running an ARM core program on the MSS. The required register values and addresses (table 4 shows examples) must first be converted into the appropriate bit stream, with the correct control bit (0: read, 1: write) added, then converted to hex for programming.

Testing of the SPI was carried out in two different ways. The first method involved direct observation of 2-way communication between the master and slave over a range of frequencies. The second method inferred correct operation indirectly through measurement of the system current draw.

Direct observations were carried out utilising the SF2 debug LEDs wired to the MOSI and MISO lines. Attaching an oscilloscope to the debug LED resistors allowed monitoring of signal quality over a range of SPI clock frequencies up to 10MHz (4 MSS). The test procedure involved writing to a register with a known value and then reading that register whilst monitoring the transaction. If the written and read data values were identical when compared the test was classified as successful.

Indirect measurements were conducted by comparing change in the system current draw while changing the number of LVDS channels. The CMV data-sheet states a power saving of 15mW per output for VDD18 supply[9], this equates to 8.3mA current per LVDS.
channel. In figure 2, the system current for LVDS combinations from 1 to 64 channels for the 3 different pixel bit depth options is shown. Extrapolating from the datasheet, the system current difference between 1 and 32 LVDS should be ≈ 260mA and the difference between 1 and 64 should be ≈ 520mA. The current increased by slightly more than these approximations (see figure 2), this was due to the nature of the testing method and the inability to isolate the CMV power usage from other components of the system. The results infer correct operation due to current increasing by at least 8.3mA per LVDS channel in the various modes.

SPI operates at fractions of the MSS clock, the fastest being \( \frac{1}{2} \text{MSS} \), and it is capped at 30MHz[9]. This presents a speed limitation on the system since gain, exposure and windowing values may have to be adjusted between exposures. This implies that the maximum speed the CMV can be programmed is 800ns, although in the current configuration with the MSS clocked at 20MHz this increases to 2.375\( \mu \text{s} \) for a single programming action (for comparison output of a whole row of 4096 pixels @ 8 bit takes ≈ 2\( \mu \text{s} \))[9]. Ideally, programming of the CMV should be minimised where possible and other methods of control especially exposure should be explored.

GPIO. The GPIO of the SF2 can also be used to control aspects of the CMV operation. The FRAME_REQ and T_EXP1 & 2 are important examples since they are used to request frames from the CMV and can also be used to control exposure time, bypassing the need to use SPI for gain control[9]. This in turn can mitigate time issues with adjusting exposure via SPI. Alternatively, FRAME_REQ can be held high for continuous operation of the CMV at the maximum rate of the current configuration. The GPIO are controlled via the ARM core program on the MSS but will operate at a rate much higher than an equivalent SPI programming action. This is due to the SPI transmission requiring 24 bits hence 24 clock cycles.

4 CMV Sensor Image Retrieval

A key milestone in this system development is the retrieval of valid image data from the sensor in the form of a picture. The SF2 does not have enough memory on-board to store an image from the 12MP CMV which, for an 8 bit pixel depth image, requires \( \approx 12\text{MB} \) of storage. The MRAM modules located on the SF2FC contain \( \approx \text{2MB} \) of RAM each with four modules arranged as 2 banks capable of storing up to 32 bits at a time[15][23]. A procedure for storage of an image was developed to facilitate retrieval of an image from the sensor and to output that image for external manipulation in MATLAB. The block diagram in figure 3 shows the basic procedural concept with a more detailed block diagram shown in figure 7.

CMV Data Processing. The CMV outputs the image data utilising DDR LVDS channels. The sensor has the capability to output data on 64 LVDS at 600Mbps in DDR with a 600MHz LVDS input clock[9]. Due to the design of the SF2FC this has been scaled down to a maximum of 32 LVDS. The 'CMV12K Interface' block in figure 1 (represented as "Processing & Interpreting" in figure 3) is the VHDL block that handles this input data. It performs three main tasks, these include: input buffering & DDR to SDR, de-serialising the pixel data and interpreting the control channel.

It is a requirement of the CMV that the LVDS input channels are terminated with a 100\( \Omega \) load resistance[9]. This is achieved by appropriate configuration of input buffers in the SF2 fabric prior to further signal processing. The input data is also DDR which, due to design factors, such as the SCoG algorithm and to reduce complexity, is not required in this application. The SF2 offers a DDR to single data rate conversion block which performs the conversion utilising latches to output data from the rising and falling edge of the clock. The output of the conversion process is the two bits, QR & QF, which change once each clock cycle.

The CMV outputs individual pixel data on a single LVDS serially, the number of bits is dependant on which pixel bit depth operating mode is chosen, either 8, 10 or 12 bits. To allow for easier manipulation of pixel data throughout the system, and faster operation, the SCoG is written to accept parallel pixel data, thus the pixel data needs to be de-serialised. This is done by stacking the incoming bits until a full pixel is received prior to producing an output. The de-serialising process is then applied to each LVDS channel, the output standard logic vector is the

Figure 3: Block diagram of basic process for taking an image from the CMV and exporting from system. A more detailed block diagram located in appendix figure 7
length of the number of LVDS channels multiplied by the camera pixel colour depth ie. \(32 \times 8 = 256\) for 32LVDS @ 8 bits.

This input processing decreases the incoming data rate from the CMV sensor since it takes several clock cycles for all of the bits of each pixel to arrive in the de-serialising block. For example, if the CMV LVDS clock is operating at 100MHz, the CMV DDR clock is operating at 50MHz and the data is arriving at the input buffers at 100Mbps due to the DDR. This implies that if the colour depth is 8 bits, it will take 4 clock cycles for a pixel to arrive at the output, equating to a data rate of 25Mbps. Operating with 32 LVDS channels this is equivalent to 28fps into the input buffer and 7fps out of the de-serialising block. The CMV DDR clock can run in the range from 50MHz to 300MHz[9] (the DDR clock is \(\frac{1}{2}\) the LVDS input clock) which implies a theoretical data rate of between 7fps and 21fps. Assuming that a centroid solution is output at this rate, at the slowest operating speed of the CMV, the system will operate at speeds comparable to contemporary systems [2][3]. It is worth noting that this is the slowest speed that is possible and it has not taken into account imaging only ROI which greatly increases the frame rate[9].

The final task this block performs is interpreting the control channel output from the CMV. The control channel is critical to deciphering the incoming pixel data, it contains information on the validity of the current incoming data, row and frame. Without the use of the control channel it is impossible to interpret incoming data incorrectly. For example, lose track of which pixel incoming bits represent thereby reading portions of two adjacent pixels. The interface block interprets the beginning of a frame by monitoring the control channel for a specific pattern of bits that represent the beginning of a frame. It attempts to do this by monitoring the incoming bits from the control channel and comparing them with the expected bit pattern. From table 3, the expected bit pattern for a valid pixel in 8 bit operation is of the form [111XXX01], therefore if the previous 8 bits match this pattern the pixel sequence is locked. The CMV control channel \(DV:AL\) (bit [0]) can then be used to control write operations.

Testing of this block, in particular the control channel decoding, is ongoing. There have been significant hurdles in determining correct operation due to the speed of operation and limitations of the hardware configuration. The minimum LVDS clock rate of 100MHz means that the signals cannot be routed to the debug LEDs in a similar way to the SPI testing method, this means direct observations of the CMV data output is not possible with the available equipment. In current testing, observations of stored data from the sensor is either not producing the expected results, by outputting ‘0’ or seemingly random data.

**Storage of Image Data.** The SF2 has limited internal storage, thus storing full 12MP images in the SF2 is not an option. It is also not feasible to capture smaller segments of an image to read out using the "smart debug" feature in Libero, since the amount of data in a 12MP image is considerable. The SF2FC has five 16x1M MRAM integrated circuit (IC) from Everspin technologies of which four are set up in two banks with common addressing and data I/O pins. Thus each bank can store 4MB of data, this means that a full 12MP image at the lowest pixel bit depth will still not fit into the MRAM, although a large portion of an image frame will. In order to then capture a full frame image there are two native solutions built into the CMV, binning and windowing. Binning reduces the image size by a factor of four since it averages that number of adjacent pixels. Windowing allows individual image rows to be selected, this means the process can be completed as many times as necessary to retrieve the image. Once the system is running the ability to store actual image data is not required, although for testing and focussing of the optics an image is useful. Two methods of interfacing with the MRAM where considered, AHB and direct control, these methods represent the bottom blocks figure 3.

**AHB.** The AHB method was explored due to the integrated memory controllers in the SF2 which remove the necessity of managing addressing manually. The integrated memory controllers are AHB slave devices and once implemented populate address locations in the SF2 memory report allowing convenient access. In order to write to the memory, a bespoke AHB master block was required, to interface with the CMV, MRAM and fabric.

The easiest way to implement the required master was to implement a state machine in VHDL, seen in figure 8. This was developed around a manufacturer application note for correct custom interfacing with MSS [25].

While the developed controller works well, with testing showing correct writing to and from the MRAM, the speed of the system becomes an issue particularly if attempting to store data at 100Mbps. Although the AHB is capable of high speeds[25], in this case it is limited by the MSS clock speed of 20MHz. The MSS clock is also not able to be increased to a speed appropriate to the storing of raw data from the CMV, due mainly to difficulties with timing and routing of internal signals. Compounding this issue is the fact that, at a minimum, the storing operation requires traversing the three write states to idle over four clock cycles. This is also assuming that the bus is de-congested the entire write operation. This implies a maximum speed, assuming fixed MSS clock speed of 20MHz, of 5MHz which is too slow for image capture and storage applications. This controller, although too slow for this application, is potentially useful for other applications in the high-speed star tracker system, such as storing centroid coordinate data.
Direct Interface. In order to overcome the AHB limitations and improve MRAM interfacing speed, a direct memory access method was employed. This involved the removal of the SF2 integrated memory controllers and AHB entirely. This bespoke write and separate read controller has the ability to overcome the negatives associated with AHB for the image retrieval task. It also decreases project complexity during testing by separating write and read operations. Due to the asynchronous nature of the MRAM, a state machine is still required to perform the transfer, since there are various hold time requirements associated with read and write operations. The MRAM also has a read and write limitation of $\approx 30\,\text{MHz}$ due to the minimum read/write time of 35ns. This implies that even if the AHB clock could have been increased, the MRAM itself would become the bottle neck. It also gives a ceiling to the potential max speed of a direct interface between the FPGA and MRAM.

Although the direct interface method still requires a state machine with at least two read and write states (see figure 4), the ability to control the clock speed of the controller/MRAM system allows the maximum potential speed of the MRAM to be utilised. This method allows for image data to be stored directly from the CMV utilising the CMV DDR clock. This is due to the DDR clock being $\frac{1}{2}$ of the LVDS input clock, in this case 50MHz output for a 100MHz input. The Processing & Interpreting block (figure 3) further divides this by at least four (for 8 bit pixel depth) when de-serialising the data. The final speed, 12.5MHz, is less than the maximum operating frequency of the MRAM and easily utilised. This method also allows up to four LVDS channels can be stored simultaneously, this is due to the MRAM I/O being 32 bits wide (four 8 bit pixels). This method exceeds the data rate required and avoids the issues with the AHB method, therefore making it the best option for storage of those explored.

Exporting Image Data to MATLAB. The process to transfer data from the MRAM and export from the SF2 is represented by the right hand side of the block diagrams in figures 3 & 7. Once the data is stored utilising either of the MRAM write methods explained above it requires reading out. Since the MRAM is non-volatile the direct method involves reprogramming the SF2 with the ‘read’ project. This separate ‘read’ project allows maximising of storage space on the SF2, thereby instantiating the largest possible CoreFIFO instance. The final step involves Libero ‘SmartDebug’ program which has the ability to read the FIFO memory locations and export them into a ‘.txt’ file for importing into MATLAB.

Testing of Image Retrieval Procedure. Initial testing of the procedure involved instantiation of a VHDL block in place of the CMV Sensor and the Processing & Interpreting block. This block outputs patterns of bits that changed through 32 pre-determined hexadecimal values before repeating. The easily identified bit pattern is recognisable (0x11, 0x22,....) in the ‘SmartDebug’ window when reading the FIFO at the end of the write/read process. This initial test procedure tests all of the blocks in figure 3 except the aforementioned two. When carried out the test procedure produced the expected bit pattern consistently, there were no observed inconsistent bits within the 2048 address deep FIFO instantiated for testing over $> 10$ write/read cycles.

Once the write/read procedure was verified in the above method, testing with the CMV Sensor and the Processing & Interpreting blocks (figure 3) was carried out. The CMV sensor has the ability to output a test signal by writing the appropriate values to registers 117 & 122[9]. This pattern is a repeating gradient from dark to light pixel values being output over the LVDS channels and is explained in detail in the datasheet[9]. The results of this test procedure were not consistent with the expected values produced by the built in test pattern. The potential issues have been alluded to previously and involve the interpretation of the CMV sensor control channel. At this stage the image retrieval procedure is not producing results consistent with the expected test pattern and further testing is required.

Figure 4: State diagram of direct MRAM writing method. This reading state diagram is identical to this with all instances of write changed to read.
B Simulation III PROJECT APPROACH

1 Centroid Identification using Matlab Approximation of SCoG Algorithm

A software approximation of the SCoG algorithm described in (Kong, Polo, Lambert, 2017)[8] was developed to aid in understanding of the algorithm prior to carrying out the FIFO replacement. The output of the algorithm, overlaid on a false colour image, is shown in figure 10. The centroid pixels are offset by \( \frac{M-1}{2} \) (seven pixels in the figure as \( M = 15 \)) down and across from the actual centroids. This is due to the convolution process adding padding around the image, although this offset adds clarity when viewing the image so remains. The figure shows that centroids have been identified for the brightest stars in the image while the fainter stars have not. This is due to the threshold setting and is used to minimise false centroid solutions from noisy images. In figure 10 the centroid threshold is set to pick up stars of \( M_v \geq 6 \) or brighter, this decision is based on the consideration of the bright star catalogue (BSC), which is a potential star catalogue for generating a look-up table[26].

The process utilised by the Matlab implementation is different to that described in (Kong, Polo, Lambert, 2017)[8] since it was implemented for understanding only and implementation on conventional CPUs as described is virtually impossible[8]. In the Matlab implementation, a convolution was performed on an image with three different filters \( F_x, F_y \) and \( F_{sum} \) of size \( M \times M \). The filter kernels \( F_x \) and \( F_y \) range from \( -M \) to \( M \) in integer increments in their respective x and y direction, the \( F_{sum} \) filter is an array of 1’s. The convolution operation utilised in MATLAB has identical output as a correlation due to filter symmetry so was utilised for simplicity.

The resulting images were “binarised” by looking for negative to positive zero crossing points in the appropriate filter direction for \( \hat{C}_x \) and \( \hat{C}_y \), and application of a threshold for \( I_{sum} \). The “binarised” images were then processed, in a manner similar to that in figure 5. First by performing an AND operation on the \( B_x \) and \( B_y \) images resulting in matrix \( S \), followed by an AND operation on \( S \) and \( I_{sum} \) with a threshold applied.

The output of the process is an array of zeros with ones located at the centroid solutions which can be further utilised in matching algorithms. Testing of the MATLAB simulation has been carried out on several images with success, a portion of a larger image is shown in figure 10.

2 Star Catalogue

There are many different star catalogues that are available, each having various differences compared to others. Several catalogues were researched in attempting to find a catalogue that is appropriate for the creation of a look-up table for a high-speed star tracker. The key factor that was considered in researching was the physics limitation associated with light travelling over long distances, basically that brighter objects will be easier to detect in shorter periods of time.

A tutorial, by Carl C. Liebe, based on star tracker design explains the mathematics for the number of photons required on an image sensor to overcome inherent noise[1]. Figure 9 shows a theoretical plot for sky coverage based on \( M_v \) and FOV. \( M_v \geq 6 \) stars and above provide full sky coverage with FOV of \( \approx 4^\circ \) which is a reasonable FOV assumption for a star tracker.

Of the star catalogues researched the Yale BSC[26] stood out due to it containing only stars of \( M_v \geq 6 \) and greater. Using a stars RA and DEC the entire catalogue was plotted on a unit sphere (see figure 11) and by using the formula \( \theta = \cos^{-1} \left( \frac{d}{\|d\|} \right) \) the nearest neighbouring stars are identified for use in triangle based star matching algorithms.
IV Conclusions

The aim of this project was to integrate and test various hardware components as well as develop the project toward the end goal of a high-speed star tracker. The initial integration of the hardware components has been completed with much of the testing carried out. The control & communication between the major hardware components has also been established, with the majority having been verified. Much of the engineering has also been completed to allow image retrieval from the CMV. Also a lot of ground work on the project has been completed, paving the way for the remainder of the design process. The simulations, developed in MATLAB, have also greatly increased understanding of stream processing of centroid data and have given a clear understanding of the intent and direction for the project.

This report has outlined the engineering process applied to the design, integration and testing of several components of what is to be a larger project, the development of a lightweight, low power, high-speed star tracker. The research and work carried out throughout the course of this project, and described in this report, gives confidence that once completed this system has potential to outperform contemporary star trackers.

V Future Work

Testing of CMV12000 and SCoG with Simulated Star Field

Once the system is receiving reliable data from the CMV sensor testing of the SCoG will need to be carried out. A simple test plan that involves a star field image placed in front of the sensor, the output centroid data is then compared to this image. This will be a simple but effective first test. A more rigorous testing method could then be employed once confidence with the SCoG is high. One proposed method utilises the high precision of the Earth’s rotation to test the star trackers accuracy, with real stars closely simulating actual performance of the star tracker in orbit. This method involves using the star tracker in a fixed position on the Earth and measuring the Earth’s movement with the star tracker. Since the Earth’s rotation is very precise the star trackers accuracy can be measured accurately[27].

Scaling up of SCoG Algorithm to Maximise Utilisation of CMV12000 Capability

The SCoG algorithm supplied is written for a single stream of parallel pixel data in a specific pattern. In order to fully realise the high-speed potential that the CMV provides (in this hardware configuration with 32LVDS channels connected) the SCoG will need to make use of the full 32 LVDS sensor outputs. In 32 LVDS the rows are read out in the pattern 1&2, 3&4,...,3071&3072 (see figure 12) with each pair of rows taking 256 pixels of readout time (20.48µs with 100MHz clock applied to CMV)[9].

How to optimise the use of the data as it comes off the sensor needs to be considered. This could include running multiple SCoG instances, with each being supplied row data as it comes off the sensor from a two channels simultaneously (1&33, 3&35, ...,31&63). Each SCoG window would then operate over a 256 pixel wide column, figure 13 shows an example. This would require 16 instances of the SCoG at which point memory requirements could become an issue.

Matching Algorithm & Database for Centroid Matching

A finalised database and appropriate look-up table will need to be created for addition to the star tracker to allow matching to take place. There are a great many resources relating to star tracker matching algorithms[28][29][30] with many comparing the mathematical complexity vs accuracy. Due to the large number of choices and the volume of information available the decision can be postponed until the majority of the system has been developed allowing a more informed decision on both the matching algorithm and the final look-up table contents.
VI ACKNOWLEDGEMENT

I would like to thank my thesis supervisor A.Prof Andrew Lambert of the School of Engineering and Information Technology at UNSW-Canberra. Whenever I had a question relating to my research and writing, his office door was always open, he was rarely actually in there, but once tracked down was always ready to stop what he was doing and assist for as long as necessary. A.Prof Lambert consistently allowed this project to be my own work inserting guidance were necessary and tempering my more lofty expectations throughout the year.

I would also like to thank my colleagues Clinton Kerr and Matthew Brougham for their input into both this report and seminar preparation throughout the semester. On top of this for being able to discuss and bounce ideas off in the first instance.

Finally, I would like to express my gratitude to my wife Casey for providing me with support throughout the years of study, culminating in this project. You have provided encouragement when required and have organised your life and the care of our child around my research and writing of this paper.

This accomplishment would not have been possible without them. Thank you.
References


REFERENCES


Appendices

Table 1: CMV12000 CMOS image sensor voltage requirements[9]

<table>
<thead>
<tr>
<th>Supply Name</th>
<th>Required Value (V)</th>
<th>Max. Range (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD18</td>
<td>1.98</td>
<td>1.80-1.98</td>
</tr>
<tr>
<td>VDD33</td>
<td>3.30</td>
<td>3.00-3.60</td>
</tr>
<tr>
<td>VDD_PIX</td>
<td>3.00</td>
<td>2.30-3.60</td>
</tr>
<tr>
<td>VDD_RES</td>
<td>3.30</td>
<td>3.00-3.60</td>
</tr>
</tbody>
</table>

Table 2: SmartFusion2 voltage requirements[19]

<table>
<thead>
<tr>
<th>Supply Name</th>
<th>Required Value (V)</th>
<th>Max. Range (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2V DC</td>
<td>1.20</td>
<td>1.14-1.26</td>
</tr>
<tr>
<td>1.8V DC</td>
<td>1.80</td>
<td>1.71-1.89</td>
</tr>
<tr>
<td>2.5V DC</td>
<td>2.50</td>
<td>2.38-2.63</td>
</tr>
<tr>
<td>3.3V DC</td>
<td>3.30</td>
<td>3.15-3.45</td>
</tr>
</tbody>
</table>

Table 3: The CMV control channel contents for the various pixel bit depths[9].

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>DVAL</td>
<td>Indicates validity pixel data on outputs</td>
</tr>
<tr>
<td>[1]</td>
<td>LVAL</td>
<td>Indicates validity of read-out row</td>
</tr>
<tr>
<td>[2]</td>
<td>FVAL</td>
<td>Indicates validity of read-out frame</td>
</tr>
<tr>
<td>[3]</td>
<td>FOT</td>
<td>Indicates sensor frame overhead time (FOT)</td>
</tr>
<tr>
<td>[4]</td>
<td>INTE1</td>
<td>Indicates integration block 1 integrating</td>
</tr>
<tr>
<td>[5]</td>
<td>INTE1</td>
<td>Indicates integration block 2 integrating</td>
</tr>
<tr>
<td>[6]</td>
<td>‘0’</td>
<td>Constant</td>
</tr>
<tr>
<td>[7]</td>
<td>‘1’</td>
<td>Constant</td>
</tr>
<tr>
<td>[8]</td>
<td>‘0’</td>
<td>Constant</td>
</tr>
<tr>
<td>[9]</td>
<td>‘0’</td>
<td>Constant</td>
</tr>
<tr>
<td>[10]</td>
<td>‘0’</td>
<td>Constant</td>
</tr>
<tr>
<td>[11]</td>
<td>‘0’</td>
<td>Constant</td>
</tr>
</tbody>
</table>
Table 4: Register address changes for 8-bit operation over various numbers of LVDS channels[9]

<table>
<thead>
<tr>
<th>LVDS Outputs/side</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reg Add</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Register Values</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sub_offset</strong></td>
<td>66</td>
<td>65535 when Disable_top = 1 and no sub-sampling else 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output_mode</strong></td>
<td>81[4:0]</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td><strong>Disable_top</strong></td>
<td>81[5]</td>
<td>0 for two sided; 1 for single sided</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setting_1</td>
<td>82</td>
<td>3618</td>
<td>2082</td>
<td>1058</td>
<td>546</td>
<td>290</td>
</tr>
<tr>
<td>Setting_2</td>
<td>83</td>
<td>5894</td>
<td>5896</td>
<td>5896</td>
<td>5896</td>
<td>5896</td>
</tr>
<tr>
<td>Setting_3</td>
<td>84</td>
<td>143</td>
<td>143</td>
<td>143</td>
<td>143</td>
<td>143</td>
</tr>
<tr>
<td>Setting_4</td>
<td>85</td>
<td>143</td>
<td>257</td>
<td>515</td>
<td>1031</td>
<td>2063</td>
</tr>
<tr>
<td>Setting_5</td>
<td>86</td>
<td>143</td>
<td>257</td>
<td>515</td>
<td>1031</td>
<td>2063</td>
</tr>
<tr>
<td>Offset_bot</td>
<td>87</td>
<td>510</td>
<td>510</td>
<td>510</td>
<td>510</td>
<td>510</td>
</tr>
<tr>
<td>Offset_top</td>
<td>88</td>
<td>510</td>
<td>510</td>
<td>510</td>
<td>510</td>
<td>510</td>
</tr>
<tr>
<td><strong>Channel_en_bot</strong></td>
<td>90-91</td>
<td>Bit 0-31 enable/disable bottom data output channels; 0: disabled; 1: enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Channel_en_top</strong></td>
<td>92-93</td>
<td>Bit 0-31 enable/disable top data output channels; 0: disabled; 1: enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Setting</strong></td>
<td>98</td>
<td>36362</td>
<td>36362</td>
<td>36362</td>
<td>36362</td>
<td>36362</td>
</tr>
<tr>
<td><strong>Clock speed adjust</strong></td>
<td>107</td>
<td>11614</td>
<td>11614</td>
<td>11614</td>
<td>11614</td>
<td>11614</td>
</tr>
<tr>
<td><strong>Vramp1&amp;2</strong></td>
<td>109</td>
<td>13416</td>
<td>13416</td>
<td>13416</td>
<td>13416</td>
<td>13416</td>
</tr>
<tr>
<td><strong>Setting_6</strong></td>
<td>113</td>
<td>788</td>
<td>788</td>
<td>788</td>
<td>788</td>
<td>788</td>
</tr>
<tr>
<td><strong>Setting_7</strong></td>
<td>114</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td><strong>Bit_mode</strong></td>
<td>118[1:0]</td>
<td>0: 12 bits/pixel; 1: 10 bits/pixel; 2: 8 bits/pixel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: Hardware implementation of a 2D symmetrical separable filter for use in stream processing of image data for centroid detection[8].
Figure 7: Detailed block diagram of for taking an image from the CMV and exporting from system.

Figure 8: AHB Master state machine design
Figure 9: Theoretical sky coverage as a function of FOV and detection threshold[1]
Figure 10: Output of centroid algorithm (purple pixels) overlaid on false colour star field. Centroid pixels are offset by (7,-7) pixels from actual centroid locations in the overlaid image.
Figure 11: Contents of the Yale Bright Star Catalogue plotted onto a unit sphere shows fairly consistent coverage over Earth’s surface.

Figure 12: Pixel readout pattern from CMV for double sided 32 LVDS channel mode[9].
Figure 13: Diagram of SCoG window moving through a 256x3072 pixel grid. As pixels are supplied to one corner of the SCoG filter window the output is provided at the diagonally opposite corner[8].