DESIGN AND IMPLEMENTATION OF A LOW-COST UAV-BASED MULTI-SENSOR PAYLOAD FOR RAPID-RESPONSE MAPPING APPLICATIONS

M. Sakr*, Z. Lari , N. El-Sheimy

Department of Geomatics Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada
(mostafa.sakr2, zlari, elsheimy)@ucalgary.ca

Commission I, ICWG I/Vb

KEY WORDS: Unmanned Aerial Vehicle (UAV), Multi-Sensors, Embedded Systems, Low-Cost Payload, Hardware Architecture, Direct Georeferencing

ABSTRACT:

The main objective of this paper is to investigate the potential of using Unmanned Aerial Vehicles (UAVs) as a platform to collect geospatial data for rapid response applications, especially in hard-to-access and hazardous areas. The UAVs are low-cost mapping vehicles, and they are easy to handle and deploy in-field. These characteristics make UAVs ideal candidates for rapid-response and disaster mitigation scenarios. The majority of the available UAV systems are not capable of real-time/near real-time data processing. This paper introduces a low-cost UAV-based multi-sensor mapping payload which supports real-time processing and can be effectively used in rapid-response applications. The paper introduces the main components of the system, and provides an overview of the proposed payload architecture. Then, it introduces the implementation details of the major building blocks of the system. Finally, the paper presents our conclusions and the future work, in order to achieve real-time/near real-time data processing and product delivery capabilities.

1. INTRODUCTION

Over the past few decades, the incidence of natural and man-made disasters has been dramatically increasing due to global climate change, infrastructure vulnerability, unplanned urbanization, and population growth. In order to efficiently handle these situations and minimize their negative social and economic impacts, development of emergency response plans is crucial. The provision of such plans is contingent on real-time access to geospatial information over the affected areas. In recent years, Unmanned Aerial Vehicles (UAVs) have been widely adopted as reliable platforms for the collection of geospatial data required for rapid-response mapping applications, due to their low deployment cost and accessibility to hazardous areas. However, the majority of these systems are not yet capable of real-time/near real-time data processing and product delivery. In order to overcome these shortcoming, this paper aims to introduce a low-cost UAV-based multi-sensor mapping payload which supports real-time processing and can be effectively used in rapid-response applications.

The main components of this payload are imaging sensors (low-cost off-the-shelf day-light cameras and laser scanners), a GNSS-INS navigation system, an onboard computing system with advanced data processing capabilities, and a two-way RF communication link. The onboard computing system integrates a tightly-coupled programmable logic fabric and a high-performance processor for simultaneous execution of real-time low-level processing tasks and high-level processing functions. The implementation of real-time tasks and high-level functions on separate processing elements significantly improves the overall performance of the proposed payload.

This multi-sensor payload has been designed to be modular, which allows for the accommodation of different imaging sensors. It is also capable of high-accuracy time synchronization between the involved imaging sensors and the onboard GNSS-INS navigation system, to time-stamp the different events and transactions performed within the payload. This accurate time synchronization procedure facilitates direct georeferencing of the captured images and/or laser scanner points. Furthermore, the proposed payload supports real-time control and monitoring by connecting to a ground control station using the two-way RF link. The proposed payload serves as a foundation for further development towards the real-time/near real-time mapping product delivery. This goal was the driving force that directed the system architecture design and the components selection. The embedded programmable-logic fabric provides an opportunity to implement specialized hardware accelerators. These accelerators could be then used to perform some of the time-consuming functions, thus off-loading the main processor and increasing the throughput of the mapping payload.

The second section of the paper describes the overall system architecture, including the target UAVs, an analysis of the effect of synchronization errors, an overview of the computational platform, the main sensors used in the system, and the wireless communication module selection criteria. The following two sections describe the implementation details of two major building blocks in the system. First, the camera interface software is described. The camera interface implemented in this work makes use of the CHDK open-source project (CHDK, 2016). Finally, the design of the GPS synchronized data logger block is presented, along with the results of using the GPS 1PPS signal to characterize the on-chip clock frequency.

2. LOW-COST PAYLOAD ARCHITECTURE

2.1 Target UAV Systems

A successful design of a UAV payload must take into consideration the restrictions and requirements of the target host vehicle (Lari and El-Sheimy, 2015). The host UAV imposes limits on the

*Corresponding author

size and power consumption of the payload. Also, the specific characteristics of the mapping missions drive the selection of the imaging components. Additionally, the range and duration of the intended mapping mission emphasize the wireless communication module range considerations. Whereas the UAV speed and dynamic characteristics define the accuracy of the georeferencing module. The more agile the UAV is, the more accurate the georeferencing should be.

Figures 1, 2, and 3 show an example of the target UAVs, that can accommodate the proposed payload. The main characteristics of these UAVs are summarized in Table 1.

The three candidate UAVs can cover short range to medium range missions, and they can operate for 15 to 50 minutes. Knowing the endurance and range of target UAV is important when selecting the suitable communication module, as will be illustrated in a following section. It is worth noting that when planning a UAV flying mission, the major limitation to the flight range could be the regulations in effect (Toth and Jkw, 2016), not just the hardware constraints of the UAV system.

UAV Type	Endurance	Range	
	min	km	mile
3DR X8+	15	<1	<0.62
Aeryon Scout	25	3	1.86
Trimble UX5	50	60	37.28

Table 1: Target UAV characteristics

Figure 1: 3DR X8+

Figure 2: Aeryon Scout

Figure 3: Trimble UX5

2.2 Synchronization Error Analysis

Synchronization error is one of the errors that contaminates the calculated coordinates of a certain point i in the mapping frame (El-Sheimy, 1996; Skaloud, 1999). This error is caused by the fact that each sensor used in the system has its own time reference. The objective of implementing a hardware-based synchronization solution is to establish a common timing reference between the imaging sensors, and the positioning solution. Then, use this reference to accurately time-tag all sensors output.

In order to define the required synchronization resolution, the effect of the synchronization error should be examined. The following analysis defines the error of the computed ground coordinates of a point i, measured using camera and lidar data, as a function of the synchronization error between the imaging device and the positioning solution.

Using camera data, the ground coordinates of the point i in the mapping frame is given by Equation 1.

$$r_i^m = r_{GPS}^m(t) + R_b^m(t)(r_c^b - r_{GPS}^b) + s_i R_b^m(t) R_c^b r_i^c(t) \quad (1)$$

where
r_i^m – ground coordinates of the point i in the mapping frame
$r_{GPS}^m(t)$ – GPS antenna phase center position
r_c^b, r_{GPS}^b – offset between the camera/GPS antenna phase center and the IMU in the body frame
s_i – point i scale factor
$R_b^m(t)$ – rotation matrix between the body frame and the mapping frame
R_c^b – rotation matrix between camera frame and the body frame
$r_i^c(t)$ – coordinates of point i in the camera frame

The coordinates error, δr_i^m, due to synchronization error can be obtained by introducing the synchronization error term, δt, to Equation 1. To derive the δr_i^m term, we expand the right hand side of Equation 2 around the correct image capturing time, t_i, and we consider the first two derivatives of the GPS position, $r_{GPS}^m(t)$, and the first derivative of rotation matrix $R_b^m(t)$, as shown in Equation 3.

Equation 4 shows the final expression of the ground coordinates error for a point i, where $\dot{R}_b^m(t_i) = R_b^m(t_i)\Omega_{mb}^b$. For traditional airborne based mapping, only the first term of the error equation is considered (Skaloud, 1999). Figure 4 shows the coordinates error for a fixed-wing UAV flight, with a synchronization error of 1 ms. The effect of the rotation matrix error, $R_b^m(t)$, is in the same order of magnitude of the velocity error. Figure 4 shows also that the effect of the UAV vibration on the final coordinates error is insignificant.

$$\begin{aligned} r_i^m + \delta r_i^m &= r_{GPS}^m(t_i + \delta t) + R_b^m(t_i + \delta t)(r_c^b - r_{GPS}^b) \\ &\quad + s_i R_b^m(t_i + \delta t) R_c^b r_i^c(t_i) \quad (2) \\ &= r_{GPS}^m(t_i) + \dot{r}_{GPS}^m(t_i)\delta t + \frac{1}{2}\ddot{r}_{GPS}^m(t_i)\delta t^2 \\ &\quad + \left(R_b^m(t_i) + \dot{R}_b^m(t_i)\delta t\right) \\ &\quad \times \left(r_c^b - r_{GPS}^b + s_i R_c^b r_i^c(t_i)\right) \quad (3) \end{aligned}$$

$$\begin{aligned} \delta r_i^m &= \dot{r}_{GPS}^m(t_i)\delta t + \frac{1}{2}\ddot{r}_{GPS}^m(t_i)\delta t^2 + R_b^m(t_i)\Omega_{mb}^b \delta t \\ &\quad \times \left(r_c^b - r_{GPS}^b + s_i R_c^b r_i^c(t_i)\right) \quad (4) \end{aligned}$$

A similar analysis can be performed to evaluate the impact of synchronization error on the ground coordinates of scanned points by a lidar system. The lidar derived ground coordinates of point i is given by Equation 5. Using the same approach described earlier, the final coordinates error, δr_i^m, considering the impact of time synchronization error is given by Equation 6.

$$\begin{aligned} r_i^m &= r_{GPS}^m(t) + R_b^m(t)(r_{lu}^b - r_{GPS}^b) \\ &\quad + R_b^m(t) R_{lu}^b R_{lb}^{lu}(t) r_i^{lb}(t) \quad (5) \end{aligned}$$

where
r_{lu}^b – offset between the laser unit and the IMU in the body frame
R_{lu}^b – rotation matrix between laser unit frame and the body frame
$R_{lb}^{lu}(t)$ – rotation matrix between laser beam frame and the laser unit frame
$r_i^{lb}(t)$ – coordinates of point i in the laser beam frame

$$\delta r_i^m = \dot{r}_{GPS}^m(t_i)\delta t + \frac{1}{2}\ddot{r}_{GPS}^m(t_i)\delta t^2 + R_b^m(t_i)\Omega_{mb}^b \delta t \times \left(r_{lu}^b - r_{GPS}^b + R_{lu}^b r_{lb}^{lu} r_i^{lb}(t_i)\right) \quad (6)$$

Figure 5 shows the impact of time synchronization error on the simulated lidar point cloud coordinates. This figure shows the effect of 1 ms synchronization error between the lidar measurements and the GPS/INS position and orientation. The simulated data utilizes the Applanix POS LV 420 GPS/INS unit and the Velodyne HDL-32E lidar unit.

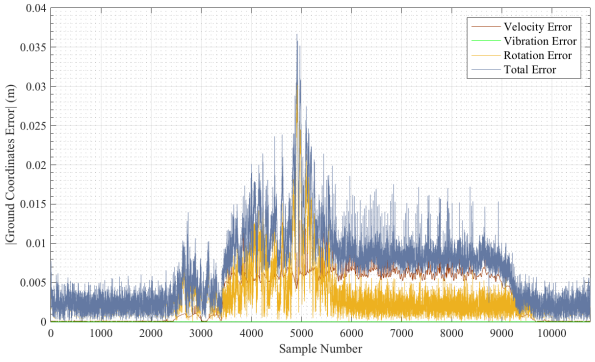

Figure 4: Norm of the image ground coordinates errors

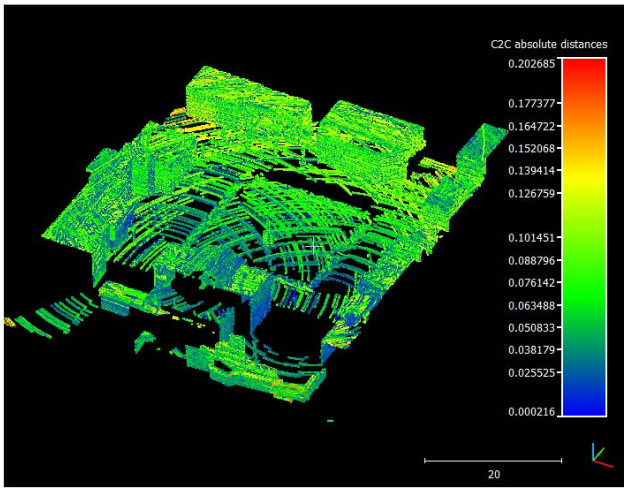

Figure 5: Simulated lidar points errors

2.3 Computational Platform

The traditional computational platforms used in several UAV systems are based on microcontrollers (Bumker et al., 2013; Mszars, 2011; Berni et al., 2009). They are low-cost, easy to program, and provide convenient I/O interfaces to connect several peripherals. However, microcontroller have serious performance limitations, that prevent them from processing large amounts of data in real-time. They are mainly used for basic data logging and time-stamping.

In order to enable the potential of real-time processing on the UAV, a more powerful and versatile computational platform is required to power the proposed payload. As a result, the proposed payload is built around the Zynq® system-on-chip (SoC), from Xilinx®. This specific SoC was selected because it provides a powerful ARM ® Cortex™-A9 processor and a tightly connected programmable logic fabric. Figure 6 shows an overview of the Zynq architecture. The Zynq platform has a lot of peripherals; such as universal serial bus (USB), Ethernet, serial-peripheral interface (SPI), I2C interface, UART and general-purpose I/Os (GPIO). The Zynq platform can connect to high speed DDR3 memory, which provides sufficient memory to run an operating system, such as Linux, and to handle large amount of data.

The flexibility of the Zynq platform comes with a challenge. That is, working with the Zynq SoC and leveraging its full power requires a steep learning curve, and requires knowledge in logic design and the corresponding logic design languages; such as VHDL or verilog. In conclusion, working with Zynq is not as easy as working with simple microcontrollers.

That said, Xilinx provides a complete design environment to facilitate the implementation of the hardware and software components of the system. The tools chain includes a hardware integrated development environment, coupled with a logic simulator, and a complete tool chain to generate the binary stream required to configure the programmable logic and the processor. Furthermore, the available tools comprise a software development kit (SDK), and a dedicated tool to configure and compile Linux operating system, which runs on the processing subsystem of the SoC.

The programmable logic devices from different manufacturers have been used in photogrammetry and computer vision applications. One example is the system proposed by (Nikolic et al., 2014), in which the authors use a Zynq-based system for simultaneous localization and mapping (SLAM) in robotics applications. The authors of (Zheng et al., 2014) use heterogeneous system, utilizing an FPGA and DSP+ARM processors, in order to achieve real-time photogrammetric processing system.

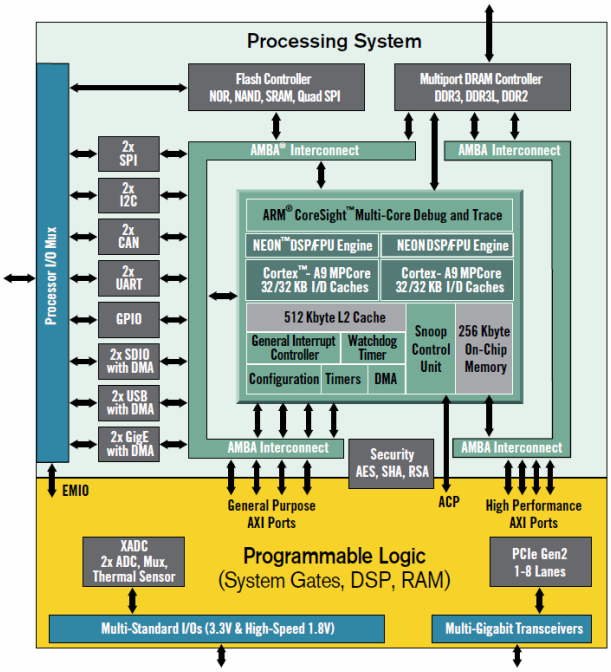

Figure 6: Zynq-7000 all programmable SoC block diagram

2.4 Payload Sensors

The proposed payload can accommodate multiple imaging devices. The payload will host two Canon PowerShot S110 cameras, Figure 7, and a Velodyne HDL-32E lidar, Figure 8. These sensors are connected to the application processor through USB

Figure 7: Canon PowerShot S110 camera

Figure 8: Velodyne HDL-32E LiDAR

and Ethernet connection, respectively. This allows for remote data exchange and control of their operation.

Each one of the two sensors has different bandwidth requirements, according to the nature of generated data and its sampling rate. Table 2 summarizes the different data types generated by the sensors, the data rate, and the total required bandwidth. These figures are an indication of which part of the generated data could be transmitted through the wireless link to the base station.

Data Type	Size	Data Rate	Bandwidth	
	Byte	Hz	Byte/s	Bit/s
JPEG Images	∼ 5 M	0.05	250 k	2000 k
RAW Images	∼ 13 M	0.05	650 k	5200 k
Point Range	3.14	700 k	2.2 G	17.6 G

Table 2: Sensors data types and bandwidth requirements

2.5 Wireless Module Selection

The wireless communication module is an essential part of the proposed payload. It is responsible for several key tasks: transmitting a subset of the captured data back to the ground control station, transmitting the system status signals to the ground station, and receiving control signals from the remote operator.

The selection of the wireless module is influenced by the required operation range and the amount of data to be transmitted and received. Tables 1 and 2 provide a summary of the operation parameters for the proposed payload. It would be ideal to find a low-cost and compact wireless module that can transmit the captured images and points to a ground base station continuously in real-time and within the entire range of the UAV. However, a survey of the available components in the market, showed that, for our target price, transmitting all captured data in real-time is not feasible. The on-board processing algorithm should define which subset of the data is sent to the ground station and which part is stored and processed on-board.

3. SYSTEM IMPLEMENTATION AND INTEGRATION

3.1 System Architecture

This section describes the overall architecture of the proposed payload. Figure 9 shows a block diagram of the payload. The system is composed of several sensors; such as GPS module, inertial measurement unit (IMU), lidar, and multiple cameras. These sensors are connected to the Zynq processing system (PS) through the carrier board, shown in Figure 11. The carrier board hosts the TE-0715-0-30 Zynq module board, shown in Figure 10. The carrier board also provides a wide array of I/O peripherals to the Zynq module. Both the carrier board and the Zynq module board are manufactured by Trenz Electronics GmbH (Trenz Electronic, 2016).

All sensors are connected to the carrier board. The two cameras are connected to the USB port, which provides the operator with full control over the camera and with access to the captured images on the fly as well. The details of the camera interface software are described in the subsequent section. The lidar is connected to the system through the Ethernet interface. The GPS and the IMU are connected to serial ports.

The GPS 1PPS signal, IMU interrupt signal, and cameras shutter synchronization signals are connected to the programmable logic (PL) section of the Zynq SoC through the PL I/O, which is referred to as SelectIO (Technical Staff, 2015). The possibility of connecting signals directly to the Zynq PL allows the implementation of custom logic blocks, to generate a reliable and a stable internal timing reference, and to log the time of the different events occurring to the connected peripherals.

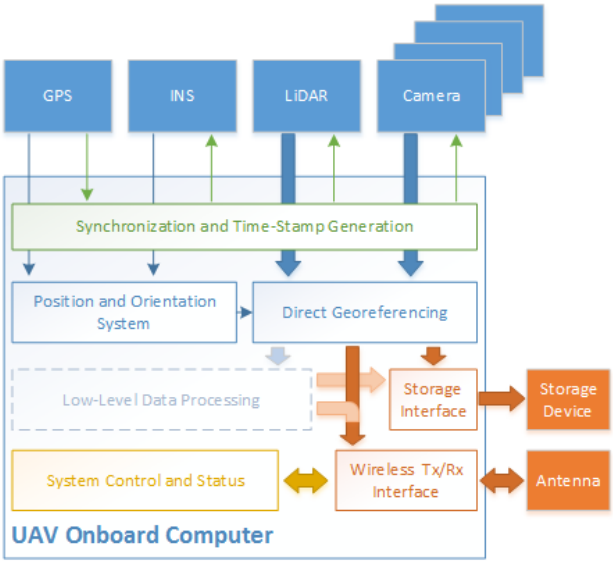

Figure 9: Payload system block digram

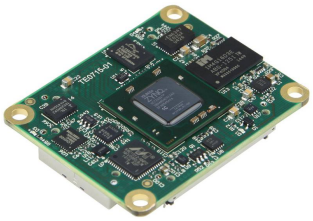

Figure 10: TE0715-03-30 Zynq SoC module XC7Z030

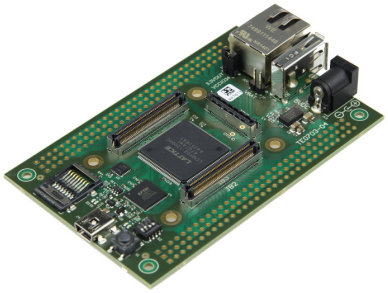

Figure 11: TE0703 carrier board

3.2 Camera Interface and Control

Figure 12 shows an overview of the camera interface. The camera interface makes us of the CHDK framework (CHDK, 2016). CHDK is an open source project, which enables advanced capabilities to the Canon point-and-shoot line of cameras. This includes enabling RAW images capturing, controlling camera parameters remotely, and many other advanced control functions found typically on high-end cameras.

Basically, CHDK is a software wrapper to the original Canon firmware. In our implementation we modified the CHDK code, to enable a more precise monitoring of the shutter operation. The modified code forces one of the LEDs on the camera body to 'ON' state, just before the shutter opens, and then forces it back to 'OFF' state right after the shutter closes. This enables more precise time tagging of the actual image capturing process, not only the time the shooting command is sent from an external control device.

CHDK provides a basic solution to enable the synchronization of multiple cameras, using the USB port of the camera. Several solutions use this feature to synchronize different cameras, in which a hardware generated signal is used to trigger the image capturing process. However, using the standard USB synchronization feature of CHDK prevents the user from accessing the data on the camera, or from controlling the camera functionality. The only way to enable and disable the USB synchronization feature is to do it manually through the camera user interface.

This limitation means that the camera cannot be controlled during the mapping mission, which significantly limits the usability of the system. Instead of using the standard synchronization feature, our approach for controlling the camera is to send an image capture command through the USB control interface, and synchronize the shutter open time, as described in the previous paragraph.

CHDK provides an external software interface, called CHDK-PTP, which allows a host device to control the camera operation and perform data transfers, to and from the camera. The payload system makes use of CHDK-PTP by integrating the open-source interfacing software to our software solution.

Figure 12 gives an overview of camera interface architecture. Each camera runs CHDK firmware on its own processor. The cameras are paired with multiple instances of the CHDK-PTP software, hosted on the Zynq PS. The master camera control software is responsible for detecting the cameras, instantiating the CHDK-PTP interfaces, establishing the connection to the camera, and relaying the commands sent by the user to the corresponding camera, and interpreting the camera response. This gives the operator a full control over the camera functionality, during the entire mapping mission.

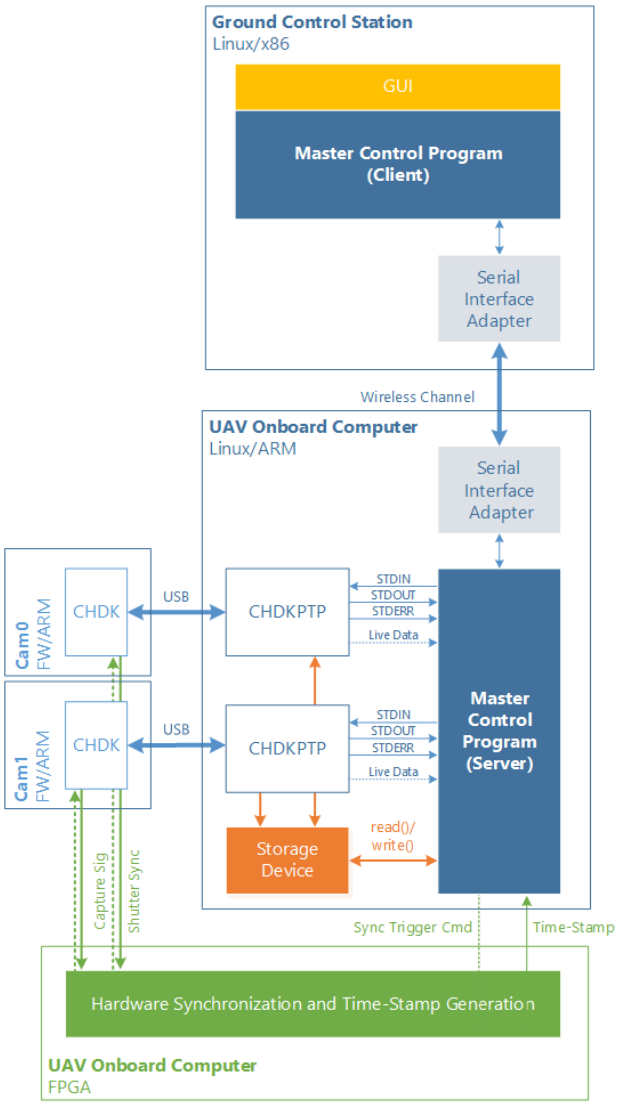

Figure 12: Camera interface overview

3.3 GPS Synchronized Data Logger

The data logging block is responsible for synchronizing the internal clock of the Zynq PL to the GPS 1PPS signal, and for time-tagging the different peripherals trigger signals. Figure 13 shows a block diagram of the data logger block. The data logger block is composed of two major sub-blocks. The first is responsible for establishing the relation between the GPS 1PPS signal and the internal high-speed clock. The 'Time Correction' block is counting how many cycles of the fast on-chip clock are there in one cycle of the 1PPS signal. This number is stored with several flags to indicate the health of the 1PPS counter. Specifically, to indicate whether an overflow has been occurred in the counter or not, and to indicate that the counter output is ready to be used by the system software.

The second part of the data logger block is the 'Time Stamping' block. This block monitors several input ports, each corresponding to a specific peripheral device. When one of these signals is triggered, the data logger block records the event time, using the high speed clock, and stores the unique peripheral ID associated with the triggered event.

The different data words are stored inside a FIFO, which is connected through a Slave AXI logic to the main AXI bus. The AXI bus provides a fast interface between the PL and the PS parts of the Zynq SoC (Technical Staff, 2015). This bus enables a software running on the ARM processor to interact with the hardware blocks implemented on the PL side of the Zynq.

Figure 14 shows the results of using the 'Time Correction' block to detect the actual frequency of the on-chip clock. The nominal value of the clock is set to 50 MHz. However, as this experiment shows, the actual clock frequency of the on-chip clock is more the nominal value by 6.17 ppm, with standard deviation of 59.52×10^{-3} ppm.

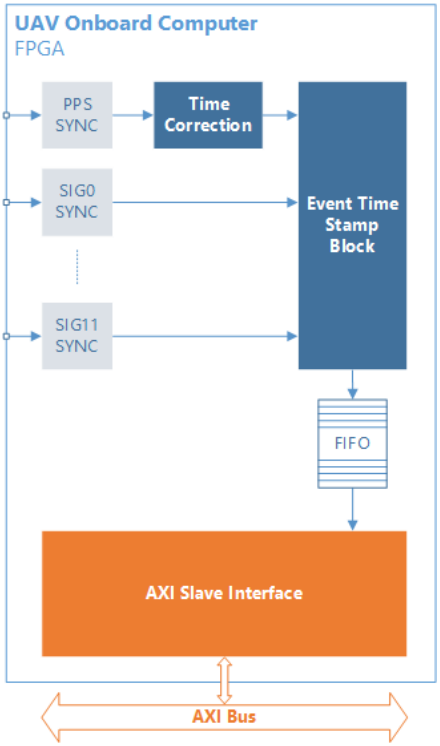

Figure 13: Data logger architecture

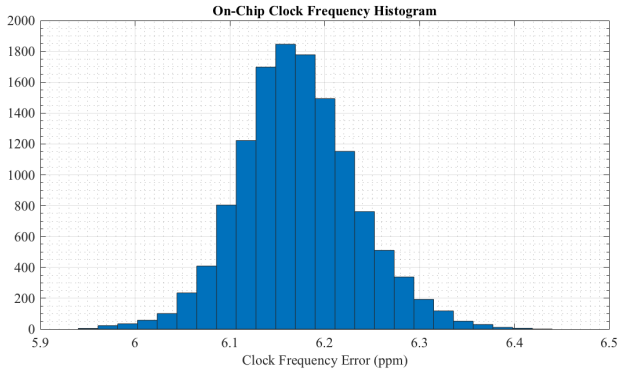

Figure 14: On-Chip clock measurements

4. CONCLUSIONS AND FUTURE WORK

This paper presents the architecture and implementation details of a UAV imaging payload geared towards the real-time/near real-time mapping applications for disasters and emergency situations. The paper describes the main components of the system, and outlines the system architecture. The implementation details of the two major building blocks of the system are presented in details; the camera interface block and the GPS synchronized data logger.

The objective of this work is to enable real-time/near real-time mapping applications. The proposed system is still a work-in-progress, and more work has to be done on different fronts, to achieve the required mapping performance:

- Migrating some of the photogrammetric algorithms to the Zynq PS. This task requires rewriting the algorithms to meet the real-time execution requirements, and adapting them to leverage the ARM processor resources.
- Breaking down the photogrammetric algorithms into parts that would be implemented as software and run on the Zynq PS, and another part that would be implemented as hardware to run on the Zynq PL. And, investigate the performance gains attained by the software-hardware partitioning.

ACKNOWLEDGEMENTS

This work was supported by Dr. Naser El-Sheimy research funds from the The Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Research Chairs programs. The data used for generating Figure 4 is provided by the Norwegian Research Council (projects no. 221666 and 223254) through the Center of Autonomous Marine Operations and Systems (AMOS) at the Norwegian University of Science and Technology.

REFERENCES

Berni, J. A. J., Zarco-Tejada, P. J., Surez, L., Gonzlez-Dugo, V. and Fereres, E., 2009. Remote sensing of vegetation from UAV platforms using lightweight multispectral and thermal imaging sensors. Int. Arch. Photogramm. Remote Sens. Spatial Inform. Sci.

Bumker, M., Przybilla, H. and Zurhorst, A., 2013. Enhancements in UAV flight control and sensor orientation. Proceedings of the International Archives of Photogrammetry, Remote Sensing and Spatial Information Science (UAV-g 2013), Rostock, Germany pp. 4–6.

CHDK, 2016. CHDK Wiki. http://chdk.wikia.com/wiki/CHDK. (1 Apr. 2016).

El-Sheimy, N., 1996. The development of VISAT-A mobile survey system for GIS applications. PhD thesis, University of Calgary.

Lari, Z. and El-Sheimy, N., 2015. System Considerations and Challenges in 3d Mapping And Modeling Using Low-Cost UAV Systems. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-3/W3, pp. 343–348.

Mszars, J., 2011. Aerial surveying UAV based on open-source hardware and software. In: Zurich: Conference on Unmanned Aerial Vehicle in Geomatics.

Nikolic, J., Rehder, J., Burri, M., Gohl, P., Leutenegger, S., Furgale, P. T. and Siegwart, R., 2014. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, pp. 431–437.

Skaloud, J., 1999. Problems in direct-georeferencing by INS/DGPS in the airborne environment. In: ISPRS, Workshop on" Direct versus indirect methods of sensor orientation", Commission III, WG III/1, Barcelona, Spain.

Technical Staff, 2015. Zynq-7000 All Programmable SoC: Technical Reference Manual (UG585). 1.10 edn, Xilinx.

Toth, C. and Jkw, G., 2016. Remote sensing platforms and sensors: A survey. ISPRS Journal of Photogrammetry and Remote Sensing 115, pp. 22–36.

Trenz Electronic, 2016. TE0715 Product Information. http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic/te0715-zynq.html. (14 Apr. 2016).

Zheng, S. Y., Gui, L., Wang, X. N. and Ma, D., 2014. A real-time photogrammetry system based on embedded architecture. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-5, pp. 633–638.