An adaptively reconfigurable computing framework for intelligent robotics

An Adaptively Reconfigurable Computing Framework for Intelligent Robotics

Moazzam Hussain, Ahmad Din, Massimo Violante, Basilio Bona

Abstract— **We present an academically developed framework for real time computations in Micro UAVs and Ground Robotics. Dynamic Partial Reconfiguration is used as a hardware accelerator in a heterogeneous environment which enables computationally demanding applications to take effective advantage of adaptive hardware resources while maintaining a high flexibility of software. We demonstrate our hypothesis by prototyping a workable application of aerial image acquisitioning and processing using partially reconfigurable hardware. We present an in-depth evaluation of proposed architecture in terms of its benefits in area, power consumption and timings.**

Keywords- Micro UAV, FPGA, partial reconfiguration, hardware platform, post disaster assesment.

I. INTRODUCTION

Micro Unmanned Aerial Vehicles (UAVs) are designed to be easily transportable platforms for rapid field deployments with very small payloads. These systems promise an easy and flexible deployment in remote areas at much reduced costs. These systems have gained a lot of acceptance in military and civil applications due to their high portability and ease of operation. Such platform could be hand carried by a single person and later operated by minimal man power using Ground Control Station (GCS). GCS are usually fitted in ground vehicles that are capable of acquiring downlinked information from Micro UAV and uplink commands for flight control, vision system control, flight mission and way points. Micro UAVs find wider acceptance in post disaster assessment in performing rapid digital photogrammetric surveys. In order to plan an effective rescue operation, it is mandatory to evaluate impact of calamity by latest geo-referenced satellite data of hit area but due to the problems pertaining to weather and satellite data availability, timely access to such information is very difficult. Micro UAVs offer an effective solution to this problem by offering an affordable and timely access to surveillance imagery of the hit areas [2]. The usage of Micro UAVs come with many constraints due to their small size and payload capabilities (which, in our case is limited to 400 grams), therefore the imaging sensors and onboard computer must be of very compact size with ultra low power consumption while must maintain their computational capability to mitigate problems related to real time deadlines, unreliable wireless links for GCS and scarce resource of bandwidth [10]. This makes the problem of designing payload electronics for such platforms two folded: first, it has to be very compact and secondly, it must be able to provide necessary functionality at very low power consumption. Therefore, single board computers are gaining more and more acceptance in carrying all the sensors and/or computing hardware [8]. These computers save space but accessibility and maintainability of such computers pose a big challenge. In order to perform maintenance, firmware update or integrity check of the hardware, the board must be plugged out of the fixture and re-mounted back upon completion of activity. In order to tackle such problems, we propose the usage of virtual hardware with Dynamic Partial Reconfiguration (DPR) of FPGA, while consistency checking/firmware updates being performed by onchip configuration manager (i.e. MicroBlaze or Power PC) when directed through wireless/serial wired link.

The main contribution of this paper is the design of a framework for configurable computing paradigm, and its evaluation on a realistic Micro UAV application followed by its prototyping on a custom hardware board with Xilinx XC2V1000 FPGA. However, the architecture is general; and is capable of supporting hardware software co-design, reconfigurable computing, real time sensor data acquisitioning/processing and performing tasks related to telemetry with a single board computer (possibly using a high-end FPGA) for ground and aerial robots. Moreover, being target to image processing, it is able to perform real time image acquisitioning and display features.

The paper is organized as follows. After a brief review of previous work, we give design description of custom developed framework followed by a description of FPGA based imaging platform. Then we present a solution to the problem of aerial image acquisitioning and processing for mosaic generation of post disaster assessment using the proposed architecture. In the last section of this paper, we present results concerning timings in multiplexing various cores through partial reconfiguration and quantitative analysis of the performance of the system. Finally, we draw some conclusions and propose extensions to the current activity.

II. PREVIOUS WORK

Traditional FPGA designs lack flexibility and are designed to implement a specific computing problem. SRAM Based FPGAs offer the ability of being partially configured at run time without affecting functionality of other working logic.

Manuscript received February 25, 2011. Moazzam Hussain, Ahmad Din, Massimo Violante and Basilio Bona, are associated with Dipartmento di Automatica e Informatica (DAUIN), Politecnico di Torino, Italy.
(e-mail: {moazzam.hussain, ahmad.din, massimo.violante, basilio.bona} @polito.it).

This feature is called Run Time or Dynamic Partial Reconfiguration (DPR) of FPGA. Run Time Reconfiguration allows dynamic multiplexing of various components in FPGA and can speed up the applications by allowing programmable hardware resources to be customized for an arbitrary fixed algorithm. The potential benefit of using run-time reconfiguration approach is obviously the significant reduction of reconfigurable resources and therefore results in reduced dynamic power consumption due to reduced chip area with added benefits of reduced cost of hardware, system level form factor optimization, enhanced performance and simple hardware design. Manet et al [5] propose a framework for DPR in imaging and signal processing applications. Compared to a completely fixed implementation, reconfigurable computing can result in reduction of FPGA device utilization by up to 50%; while in term of performance, as compared to a fully software implementation, the run-time reconfiguration approach is over 30 times faster [13]. DPR allows efficient implementation of computationally complex algorithms that otherwise will not fit the given smaller FPGA. Through timesharing of computational resources in smaller segments of a demanding algorithm, and, by instantiating less hardware in a smaller FPGA the length of critical path in the digital design can be reduced. Such implementation guarantee reduced dynamic power consumption while there is a slight rise in power consumption during the process of partial re-configuration [4], however this time interval is significantly less than the overall execution time of the algorithm. Despite the promising benefits of dynamic partial reconfiguration, its effectiveness is blotched by complex system development process and limited tool support [1].

The ability to perform high speed computations under stringent constraints of power and limited form-factor of platform is addressed by many researchers. Henrik et al [8] propose a hardware platform of a credit card size that implements a small flight control system for UAVs with efficient sensor interfacing using FPGA/DSP technology and offers computational power upto 2 GFlops/sec. [9] uses soft-core Nios processor for pan-tilt control of a camera deployed with UAV that can keep camera fixed on a specific angle using a gyro stabilized platform for vision and tracking. [10] propose a lightweight FPGA based object identification and tracking for Micro UAVs that adds to autonomous flight by enabling the onboard computer to decide, based on the information gathered from environment in addition to uplink from ground. This feature is particularly helpful in case of unreliable downlink or hostile environment with unknown terrains. Osamah et al [11] propose a reliable computing framework with reconfigurable implementation of avionics and control computers based on "graceful degradation model". [12] propose a novel autopilot description using FPGA based parallel processing for signal conditioning. [6] address real time FPGA implementation of feature detection and tracking for Micro UAV systems as a solution to onboard obstacle avoidance using feature based matching. During the flight of Micro UAV, its motion is modelled by an Affine Model [7] and therefore the need of high speed computation of Affine Transform is undeniable. A contribution in reconfigurable, intelligent co-operative robotics based on Dynamic Partial Reconfiguration has been made by [20], where a case study of individual configurability and task level re-organization at run time is implemented in a team of robots to facilitate flexible, efficient and fault tolerant implementation of complex behaviours.

III. FRAMEWORK

This section describes our efforts in designing a custom framework that helps rapid development of single board computers in robotics. This framework is envisioned as a design-base for creating different robotics applications by simplifying the generation of hardware based architecture that require minimal design and verification effort from the user. This framework is designed in view of the fact that most of the robotics and algorithm designers confront a learning curve while working with electronics system design and low level HDL programming. Although through the usage of Celoxica, Handle C, Impulse Co-developer or Matlab, now it is possible to automatically generate HDL code for a specific FPGA but there is almost always a huge margin of improvement in terms of frequency and chip area. Our proposed architecture is designed to enable a mechanical system to fly and/or navigate with minimal hardware design effort.

The development of partially reconfigurable framework is motivated by the fact that onboard computers usually work in a controlled loop in autonomous systems. For example, if onboard inertial sensors update attitude information of vehicle (involving roll, pitch and yaw) after every 20 ms and given highly parallel architecture and high speed computation (being completed in 3ms), the navigation controller must wait for 17 ms for new data. During this interval, the hardware resources could be rescheduled for another computationally demanding algorithm and when needed, the mission computer hardware could be loaded through configuration interface of FPGA with minor timing penalty. The execution of previous computations could be resumed upon completion of navigation related tasks. This framework is particularly suited to Micro UAVs and ground robots having severe limitations in terms of payload electronics size or cannot afford deployment of computationally capable hardware due to stringent power budgets.

The growth in computational capability of FPGAs and bandwidth of onchip buses is the other motivation for the proposed framework. In modern FPGA devices like Xilinx Virtex- 4/5/6 FPGAs we can obtain data rates of multiple Gbits/sec with on chip buses. Partial Reconfiguration of FPGA is performed using ICAP interface. In Xilinx Virtex-II Family, the ICAP has a width of 8-bits while in modern families it reaches to 32-bits and enables DPR with exchanged data rates reaching upto 3 Gbits/s at 100 MHz [5], this reduces partial reconfiguration time exponentially (when compared to Virtex-II) and is particularly suited to robotics and UAVs.

To our knowledge very few frameworks for mobile robotics have been proposed that offer ready to go modules with Plug n Play features based on DPR. Furthermore very few hardware architectures target real time computations on single board FPGA based computers with fully parametrizable cores for generic robotic architectures (generic with run time power management features).

A. *Framework Architecture*

The proposed framework consists of many standard and custom pre-verified reusable IP cores (in Verilog and VHDL) that can rapidly be integrated through automated scripts and can be simulated and synthesized to given FPGA technology. The usage of pre-verified IP offers an added advantage in prototyping the resulting system on FPGA. As Chipscope cores cannot be inserted in a partially reconfigurable region [5], using pre-verified IPs, the designer is relieved from in-chip debugging and if needed, interface signals may be evaluated for their correct functionality. The generic (fully parametrized) nature of cores for navigation controller, sensor data acquisitioning and imaging enable them to be deployed with a wide range of actuator/servo or vehicle dynamics. This framework is designed to be extendible to match specific needs of user. Considering the nature of target applications that work under severely limited budget of power, we implemented efficient power management strategy based on clock gating. The system runs on a system clock that clocks most of the hardware sub-systems, and CPU clock that clocks the onchip RISC. Each component is wired in a way that its input clock can be disabled to reduce power consumption and is controlled by the RISC Processor. The FPGA region(s) under DPR is also gated through bus macros to implement a power save mode as programming a blanking bitstream will not reduce power consumption in the region [4].

Given the structure of Reconfigurability, the static part usually consists of interface specifications like Ethernet/ Ethercat, GPIO, Chip Bus Architecture, Serial interfaces, SDRAM/SRAM/Flash Memory Controller, VGA, Timers, DMA, watchdog, DSP (TI TMS320C64x) interface, Flash Interface, Interface FIFOs etc, and a configuration manager (usually a RISC Processor). The computationally demanding payload cores like navigation computer, way point detector and mission computer, image/data compression, Image transformations, feature matching engines, Kalman filters and many more are implemented as a Dynamically Reconfigurable Blocks. It may be noted that each peripheral has a local FIFO that buffers single set of data to be read by the reconfigurable core when needed. As this framework is under development, we are still working on design and development of many of reconfigurable IPs. The block diagram of overall system with currently used case is shown in Figure 1. The description of each IP in the framework would require a lot of space and is beyond the scope of this article.

Run time slot time allocation is performed by the configuration manager by partially reconfiguring specified regions in FPGA that guarantees effective utilization of on-chip resources. This architecture offers an uncanny ability of "mission change" on the fly that may be used in implementing evolvable computing and complex behaviors in co-operative robotics. When directed by the GCS or local controller, the mission (the sequence and type of deployment of partially reconfigurable cores) could be modified according to the changed needs. For example, upon receiving an interrupt for immediate autonomous landing, the configuration manager must remove image compression cores from the schedule of timeslots and must schedule navigation controller with vision based tracker to implement autonomous landing. Likewise, GPS processing core must be scheduled after 1-second according to the update rate of GPS data and during the intermittent interval, the resources occupied by GPS computational core may be allocated to other algorithms.

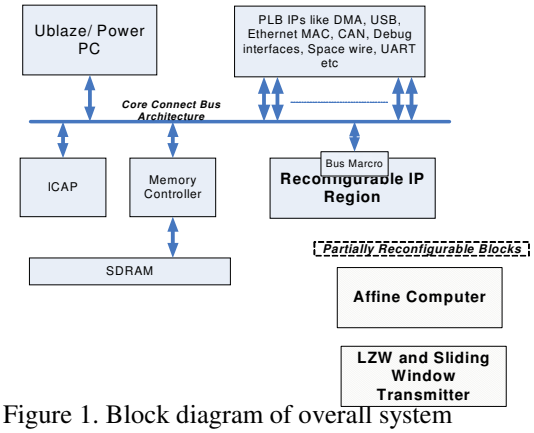

Figure 1. Block diagram of overall system

B. *Design Process*

For the development of this framework, we used Xilinx ISE with Early Access Partial Reconfiguration Tools and Plan Ahead. For the end user, the usage of this framework is simplified by its GUI that hides unnecessary details from the end user. The user is allowed to select the static part IP set, and collection of IP cores to be used in Dynamic Part. The framework already embeds the partial bitstreams of reconfigurable blocks with flexibility of being deployed to either of the two already developed floor plans shown in Figure 3.

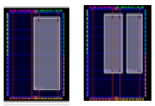

Figure 2. Two Floor Plans for XC2V1000

In order to incorporate these partial bitstreams in the final hardware, the user must generate flash storage file with a collection of partial bitstreams. This file is programmed in the onboard flash before the start of mission using onchip RISC. Once the user clicks "Generate Scripts" option in the static configuration region, relevant HDL files, NGCs and bus-macro files are copied in a directory structure as proposed by Xilinx to be used with PlanAhead for DPR and top wrapper is automatically generated with instances of all the selected static set IPs, bus macros and reconfigurable

blocks. The top level script invokes Tcl scripts of Xilinx PlanAhead for Partial Reconfiguration Flow and full/partial bitstreams are generated in the assigned directories, ready to be deployed with the hardware.

IV. Design of RIS

Nowadays, the mainstream design template foresee the adoption of an embedded processor, running an application software, coupled with specific hardware modules to accelerate the computations that are particularly time-critical, and to provide versatile input/output interfaces. Therefore, novel prototyping platforms should thus provide processor core with sufficient program memory, adequate reconfigurable resources, flexible set of interfaces and a complete software model of system for efficient simulation [3]. The ideal platform must support implementation of design features, interconnectivity of IP blocks and must have enough resources to finalize the refinements in the design and get them fit in the programmable chip. Considering our very special needs of diverse sensor interfacing, high computational capability and ultra compact size; we envisioned a real time FPGA based platform, capable of frame acquisitioning from camera link and execute computationally intense tasks in real time with a wide set of interfaces. These requirements were materialized with the development of new low-cost platform named RIS. This board may also be deployed in industrial and medical display market, cryptography, control, high speed interfacing and many more. The form factor of the board is especially suited to the limited space in COTS Robotic Platforms and Micro UAVs.

The board is equipped with 4- serial ports based on RS-422, one Giga-bit Ethernet Interface, Four optically isolated digital I/Os, 32- MB (32-bit wide) SRAM and two way connection for frame acquisitioning decoder using National DS90288 and control for camera link using DS90287 ICs and 6-Mounting holes. The data received from camera is buffered in a FIFO, and written to SRAM through DMA Controller in real time. If needed, the computational capability of the platform can be scaled using external DSP daughter card that performs read/write operations through its EMIF-A and FPGA Pins using 100-pin I/O connector. This connector could also be used as a wide bus general purpose I/O. Camera Link connector is carefully placed for high speed LVDS connectivity and to get fit easily into a mechanical enclosure with proper impedance matching of high speed traces on board. The speed of the connectivity between external DSP processor and the FPGA depends upon their mutual distance, electrical characteristics of the connector and the boards; which directly affect the setup and hold uncertainty windows. The connection has been successfully validated to establish a high speed connection of upto 1-Gbits per second between the elements using 64-bit EMIF-A. Several push button and DIP switches are available for user interaction with the system and the user can also use the VIO cores of Xilinx chipscope to generate the control signals during the upbring process.

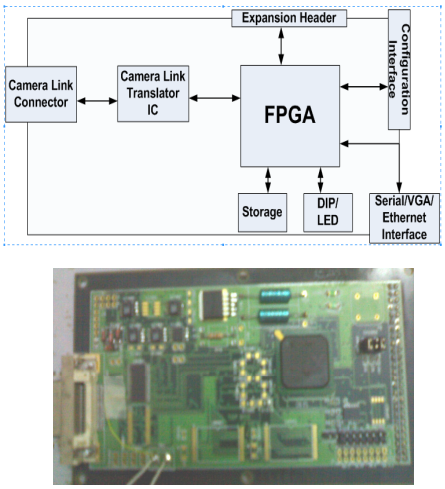

Figure 3. Block Diagram and Image of RIS during Up-bring

V. Implementation of Disaster Assessment and Aerial Data Acquisitioning/Processing

The acquired images must be transmitted from Micro UAV to ground station so that a mosaic of the images could be generated to assess level of destruction to infrastructure and property. The transmission of a full image (from Bobcat Imperx Camera) would require a bandwidth of (1000x1000 pixels with 8-bits per pixel) 8-Million bits per image. In our case, we plan to transmit at least 4 Frames per Second (FPS). Incorporating such a high speed transmission in a Micro UAV would surely deplete onboard batteries in a matter of seconds with added difficulties of portability of wireless transmitters. We propose to perform geometric alignment of images at lower resolution using onboard reconfigurable hardware and mosaic generation at GCS. Using the proposed framework from Section-III, we developed a partially reconfigurable geometric transformation hardware core. The output of the core is 100x100 geometrically aligned images having 8-bits per pixel with minimal loss of information. This is followed by LZW type compression in hardware. These techniques enable us to send sequence of compressed 4-FPS (including the overheads of sliding window control) to GCS at the bandwidth of 115 Kbits/sec and deployment of low power wireless transmitters.

In order to generate a transformed image from the acquired images and vehicle attitude information, we used forward mapping with geometric transformations followed by bi-cubic interpolation that uses input image to generate spatial positions and intensity values of target image. These geometric transformations include Scaling and Rotation in all three axes to compensate vehicle attitude. Camera parameters like CCD resolution, CCD diagonal size (in meters), focal length of the lens, vehicle altitude (height from ground) and attitude (Roll, Pitch and Heading) are used to compute resulting transformation of the image and scale factor between acquired aerial and satellite or assumed target reference image resolution. The non integer downscaling in both axis of the (UAV) acquired image is implemented using

the following transformation (for homogenous co-ordinate system).

$$T_{1/s} = \begin{bmatrix} 1/S_x & 0 & 0 & 0 \\ 0 & 1/S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The roll, pitch and heading attitudes of the vehicle are available from onboard Inertial Measurement Unit (IMU) through serial interfaces and this information is updated every 20 ms in our case. The usage of MEMS based ultra compact IMU offer reduced power consumption and smaller form factor yet accumulate more errors with the passage of time. As we acquire images sequentially and run camera under external trigger mode, we save the latest vehicle attitude parameter stamp with the current image. We used GPS as an absolute position estimation method and tied GPS information along with IMU data to every transformed image.

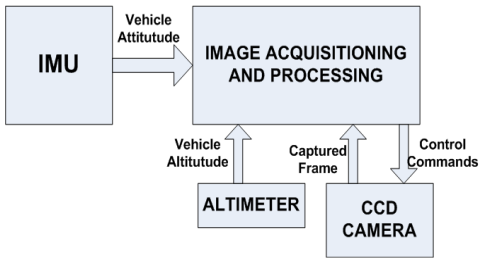

Figure 4. Block Diagram of Sensor Integration for Imaging

Since Micro UAV is subjected to 6-degrees of freedom and its attitude may be different for every image, we need a model to transform the camera co-ordinates into world co-ordinates which in our case are assumed to be ENU (Local East North Up or local geodetic) with East along X-axis, North along Y-axis and height along Z-axis. We used angles from onboard IMU to model transformation between earth and Micro UAV co-ordinates and to compensate for misalignments of CCD plane against the world plane. These angles were used in 3D homogenous geometric rotation transformation.

Due to the roll and pitch angles the acquired image will contain distorted 2D perspective that has to be adjusted while heading angle is taken as the angle from North and is compensated accordingly. We use Θ_p, Θ_r, and Θ_h as the pitch, roll and heading angles respectively. The rotation matrices of the vehicle attitude are computed as:

$$R_{pitch} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_p) & \sin(\theta_p) & 0 \\ 0 & -\sin(\theta_p) & \cos(\theta_p) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{roll} = \begin{bmatrix} \cos(\theta_r) & 0 & \sin(\theta_r) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_r) & 0 & \cos(\theta_r) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

From the above transformations, a combined transformation is generated that compensates for perspective and rotation one by one. The FPGA implementation of proposed transformation requires rapid computation of trigonometric functions and high speed access to onboard storage. The hardware building blocks of the image transformation consist of high frequency single cycle cordic core, DMA to access onboard RAM with burst transactions, fixed point arithmetic units and onchip FIFOs. The resulting transformed image is subjected to high speed data compression using custom developed LZW compression engine. This core is fully parametrized for variable sized dictionary storage that implements memory efficient hardware. The timings of these systems are given in the next section.

During the synthesis of partially reconfigurable blocks (image affine module and LZW compressor) we learnt that each module occupies less than 30% of XC2V1000 FPGA and are suited to floor-plan with two smaller reconfigurable regions for reduced sized bitstream. As the current implementation does not require any other computation related to vehicle guidance or navigation, we implement a very simple slot scheduler application with rapid partial reconfiguration of FPGA using MicroBlaze processor. We implemented a power save modes in one of the reconfigurable region permanently as it is not being used in the current design. The other reconfigurable region switches between image geometric alignment and compression modules and is governed by onchip RISC. Similar power save mode is implemented with static region IPs, so that they should be enabled only in specific intervals or else should stay in power save mode. Overall computing resources may be put to a halt state upon detection of overflows in wireless transmission link due to uncertainty in wireless links as managed by a sliding window control. Implementation of flash interface in the static part of FPGA can help storing transformed images in the onboard flash. The speed of these writes depends heavily on width of flash interface and writing agility of the flash chip. This data is retrievable by the GCS through wireless connection or wired (serial) link by giving commands to onchip RISC.

VI. Post Processing of Received Images

Image mosaicing is used to increase field of view (FoV) by stitching many images together to obtain a single image. Automatic Image mosaicing is a challenging task and require a range of tasks, like alignments of images both at global and local levels, feature extraction, feature matching, etc. Feature based image mosaicing is one of the widely used method in building mosaics because they are robust against larger disparities. In these approaches, a set of corresponding feature points are selected on the two images and homography is estimated using these points only. Images sent by Micro UAV are north aligned and affine transformed, so view point and scale doesn't change significantly. In such situation Harris corner detection [15] is a good option because of its ease of implementation, and robustness against noise and illumination [16] than SUSAN [17], SIFT [18] and SURF [19], that in addition require higher computational power and memory. Harris corner detector extracts feature points from two images. Rotations and orientations are

determined so that images can be matched. Homography was estimated by nonlinear method of RANSAC. The hypothesis of correct alignment was verified by comparing probabilities of RANSAC generated inliers and outliers for a matched and mismatched image. We compute the number of inliers and number of features in the overlapped region and performed thresholding to classify the two images as correctly aligned or otherwise. Using these estimation images mosaic is generated.

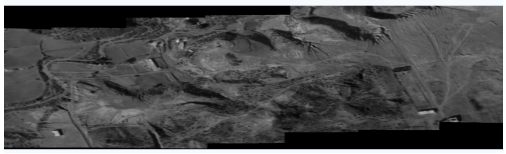

Figure 5. A sample mosaic of acquired aerial images

VII. RESULTS

The proposed framework is prototyped on a custom platform and its efficiency/affordability is evaluated in a reconfigurable environment. Both the runtime applications of image data processing and compression are evaluated for their chip area (using XC2V1000 FPGA) and timing. Table-I show the size of partial bitstream (without any pre-defined floor plan) and time to complete execution at 80 MHz for single aerial image.

Application	**Bitstream Size**	**Exec.Time@ 80 MHz (Clock Cycles)**
Image Transform & Interpolation	178 K Bytes	3 ms(0.26Million)
LZW Compression	146 K Bytes	2 ms (0.15 Million)

Table-I: Bitstream size and Time for Reconfigurable Applications

The cycle count of image transformation and interpolation filter remains almost the same for all the images, while the cycle count of LZW compression fluctuates in wide ranges as higher compression would result in more memory accesses. Maximum numbers are reported in Table-I.

The partial bitstream could be loaded in the configuration memory of FPGA through JTAG, select map or its subset ICAP in RIS hardware. In proposed architecture, ICAP is driven by onchip configuration manager. The in-circuit DMA is designed to fetch bitstreams from external storage at high speeds. Therefore, bottleneck may arise in speed of fetching data while working with bigger bitstreams [5]. A timing analysis in terms of "time to reconfigure" the hardware (for partial bitstream covering 30% of XC2V1000 FPGA in Floorplan-2) through various configuration interfaces is shown in table II:

Type of Configuration Interface	**Time for Partial Reconfiguration**
Parallel-4	470 ms
Platform USB	320 ms
ICAP using MicroBlaze @ 40MHz	3 ms

Table II: Time to Reconfigure with various Configuration Interfaces

Now we evaluate the overall timings of the system.

Application	**Time to reconfigure**	**Frequency of occurance**	**Time to complete execution**	**Available Slack**
Attitude acquisition	Static Part	50 Hz	1 ms	19 ms
Image Affine	3 ms	4Hz	3 ms	244 ms
LZW	3 ms	4 Hz	2 ms	245 ms

Table-III: Overall Timings of the System

Upon Power Up, full bitstream with static blocks and image affine and interpolation filter is loaded in FPGA. As given in Table-III, the context switch between imaging and compression filters is performed at a frequency of 4Hz. A detailed timeline is shown below:

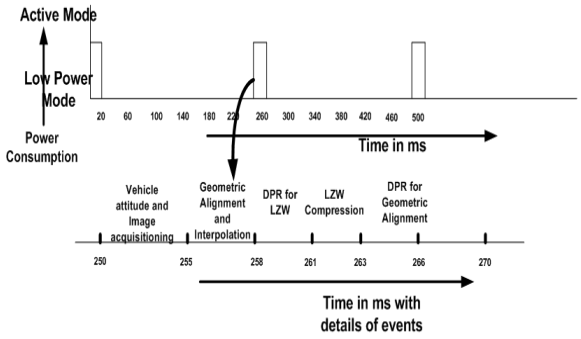

Figure 6. Timeline of Events

If we assume a minimum timing penalty in state transition of logic blocks from power save to active mode or vice versa, then it is obvious from timeline that the hardware remains operative only during discrete intervals (10% of the overall time) and otherwise it remains in the power save mode. Similarly, the static part is active during the interval of acquisitioning and processing of images otherwise it remains in power save mode. Secondly, the proposed framework architecture increases the computational capability of an old FPGA device (XC2V1000) to enable it to run a demanding algorithm that would occupy upto 75% of FPGA in a tranditional design flow while still maintaining a power save mode for more than 80% of the time.

The amount of compression on transformed images depends upon the statistical parameters (like variance) in the images, the lower the variance the higher is the compression ratio is; and we got a range of compressed sizes ranging from 0.6 K Bytes to 8 K Bytes per image. Compared to the original size of the acquired image from camera before geometric alignment, there is almost a 100-times size reduction with minimal loss of information. This loss is further compensated by acquiring multiple overlapped images per second. It is important to note that given transformation and compression techniques enabled efficient data transmission through low-cost wireless link.

An evaluation of power may be helpful in quantifying the power saving during the operational life of the system. In [4], it is clear that power consumption of the FPGA device rises during the process of partial reconfiguration roughly by 30%. As given in the timeline, we need to perform partial reconfiguration only 8-times per second for 4-images and this time is 5% of the total time. Therefore, we conclude that despite the higher cost of power consumption during partial reconfiguration, we are able to save much more power due to power save modes of the device while still meeting real time deadlines and maintaining hardware complexity to a minimum.

VIII. Future Activities

This framework is still under development and the proposed implementation is the very first practical application of this framework. Once we have developed enough RTL for IP blocks, we plan to publish it as an open-source hardware library for intelligent mechatronics to enable a continued growth of the framework. In the near future, we plan to perform a thorough study of power analysis (Watts per computation), further form-factor reduction in future platforms and re-engineering for latest FPGA technologies. An other direction of work is to modify the base platform to LEON3 [14] and AMBA buses due to their open-source nature and adaptability to a wider range of FPGA manufacturers. As the complexity of the proposed system grows, we need to develop more intuitive application for resource (time slot) management of configurable resources. This may even lead to use a small footprint operating system. We plan to develop an Automated Software Framework that generates target software application by acquiring number of timeslots and time allocation for each slot from the user and further simplify the development of application.

IX. Conclusion

Configurable computing can be seen as hybrid of ASIC and programmable processor. FPGA is ideally suited as platform for configurable computing due to its ability of getting dynamically structured in hardware according to the need of the algorithm. For many demanding applications like image processing, configurable computing can surpass alternative solutions with much less power consumption. The combination of dedicated hardware with fast configurability is an interesting academic research topic and we presented a framework based on reconfigurable computing prototyped on low-cost but capable research platform architecture. Then, we present an example implementation of image transformation and compression on a reconfigurable SOC in power and area constrained environment of Micro UAV.

X. References

[1] Miguel L. Silca et al, Journal of Systems Architecture: the Euromicro Journal, Support for partial run-time reconfiguration of platform FPGAs, EURASIP Journal on Embedded Systems (Vol-8, issue3, Article No. 16) 2008

[2] Benda H., Boccardo P., et al, Low cast UAV for Post-disaster assessment. ISPRS, 2008 Page(s) 1373- 1380

[3] Patrick I Mackinlay1 et al, Riley-2: A Flexible Platform for Codesign and Dynamic Reconfigurable Computing Research. FPL 1997. Pages: 91-100

[4] Hussain, M. et al, Power analysis of hardware based motion estimation in a heterogeneous reconfigurable environment. ASQED-2009 Pages: 325-329

[5] Philipe Manet, Daniel Maufroid et al, An evaluation of dynamic partial reconfiguration for signal and image processing in professional electronics applications. Eurasip Journal on Embedded Systems, 2008

[6] Tippetts, Beau et al, FPGA Implementation of a Feature Detection and Tracking Algorithm for Real-time Applications, Advances in Visual Computing,LNCS, Springer press, pages 682-691, 2007

[7] David, L., Video Stabilization and Object Localization Using Feature Tracking with Small UAV Video. PhD thesis, Brig. University (2006)

[8] Henrik B. Christophersen et al, Small Adaptive Flight Control Systems for UAV using FPGA/DSP Technology, AIAA 2006

[9] Ole C Jackobsen and Eric N Johnson, Control Architecture for UAV Mounted Pan/Tilt/Roll Camera Gimbal. AIAA 2005

[10] Price, A. et al, Real time object detection for unmanned aerial vehicle using FPGA based vision system. ICRA 2006, Page(s): 2854-2859

[11] Osamah A Rawashdeh, et al, A UAU test and development environment based on dynamic system reconfiguration. WADS 2005, Pages 1-7

[12] Alvis, W. et al, FPGA based flexible autopilot platform for unmanned systems, Control & Automation, 2007. MED '07, pages.1-9

[13] Nikolaos S. Voros & Konstantinos Masselos, System Level design of Reconfigurable System on Chip. FPL 2006. Page: 1-6

[14] www.gaisler.com

[15] C. Harris and M.J. Stephens, A combined corner and edge detector. In Alvey Vision Conference, pages 147–152, 1988.

[16] C. Schmid, et al, Evaluation of interest point detectors. International Journal of Computer Vision, 37(2):151–172, June 2000.

[17] S. M. Smith, Brady J. M, SUSAN- A New Approach to Low level image processing", International Journal of Computer Vision, 23(1): 45- 78,1997

[18] DG. Lowe, Recognition from Local Scale-Invariant Features, Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, Speeded- Up Robust Features (SURF). Comput. Vis. Image Underst., 110(3):346–359, 2008.

[20] Commuri S, et al, Task-based Hardware Reconfiguration in Mobile Robots using FPGAs. J Intell Robot Systems 49: Page(s) 111-134, 2007