SSC21-VII-04

On-the-Fly Hardware-Accelerated Image Processing System for Target Recognition

Eugenio Scarpa, Emilio Fazzoletto
Argotec SRL
Via Cervino, 52, 10155 Torino, Italy; +390117650567
eugenio.scarpa@argotecgroup.com

ABSTRACT

Upcoming robotic exploration missions are characterized by constantly increasing spacecraft autonomy requirements. This approach includes the necessity to face new challenging tasks, such as autonomous navigation capabilities, adaptive activity scheduling and on-the-edge data processing. When these scenarios meet small satellite platforms, coming with a plethora of resource constraints, an optimized implementation of computing intensive functionalities is necessary to achieve usable performance. Argotec, an Italian aerospace company, designs and develops its own avionics systems to enable challenging inter-planetary small satellite missions. Within this context, Argotec developed a proprietary implementation of a high-throughput image processing pipeline to support vision-based autonomous navigation and attitude control.

The purpose of this paper is to present the functionality and the performance of this system. Recalling the building blocks being in the image processing chain, the paper starts by listing these blocks and their functionalities, discussing the reasoning behind their inclusion. These functionalities include data binning, low-pass filtering for edge smoothing, color depth compression, binarization, luminance histogram generation, and eventually multi-target labeling. The challenges of delivering the required performance, high-enough to sustain on-the-fly processing in couple with state-of-the-art space cameras, are presented through the step-by-step integration in a flash-based space-grade Microsemi RTG4 FPGA.

The hardware implementation was intentionally generated to be parametrizable and platform-independent, allowing for operativity extensions, scalability and general portability. The datapath was conceived to keep functionalities as separated black boxes, each one autonomically operating. Every functional element expects processed pixels as input from the previous module and generates outputs for the following one. This solution allowed to succeed in an about 20 times faster SW-implementation running on a 50MHz Space-grade SPARC V8 processor, with a very low resource occupation in the FPGA device. In this context, improvements led to a drastic processor utilization time unloading, leaving additional place for extra tasks during mission control cycle period.

The technology is eventually analyzed in the real-life application of the DART/LICIACube autonomous planetary defense mission, proving how the design supports the mission-specific pipeline deployed for the critical NASA mission. The paper includes a final consideration that reflects on how technologies related to autonomous navigation are critical for small satellite platforms, and nowadays this aspect is calling for the need to design and tailor new solutions. This image processing pipeline wants to be an example of how new solution can enable multiple mission scenarios, until now considered prerogative of larger platforms.

1. CONTEXT

Image processing algorithms are to be largely found in the field of hardware (HW)-accelerated functionalities for multiple applications [1][2], including avionics for Space [3][4]. The exploitation of hardware resources allows for reduced time penalties when reporting the elaborated results to software (SW)-level procedures [5][6]. Minor the latency to extract the image features, minor the time for managing system control operations. In this context, autonomous navigation and attitude control are particularly suited for HW-implemented image processing algorithms [7], such as target recognition, performed in the LICIACube (LCC) satellite [8][9], developed by Argotec for the NASA planetary defense DART mission. The satellite integrates a primary Optical Payload (OP1) module which interfaces with the On-Board Computer & Data Handling (OBC&DH) unit [10] Microsemi RTG4 Field Programmable Gate Array (FPGA) [11] via a SpaceWire (SpW) interface. Images pixels are serially received and handled into a processing Image Subsystem (IS) described in VHDL and delivered into the FPGA. The IS consists of a high-throughput (up to 50Mpixels/s) pipelined datapath that sequentially transforms raw pixel data, delivering an encoded image within which each

target is labeled differently. The results are transferred to an on-board Synchronous Dynamic Random-Access Memory (SDRAM) buffer controlled by the FPGA itself and accessed by SW routines at the end of the elaboration.

2. IMAGE PROCESSING FUNCTIONALITIES

Background and purpose

The LCC satellite OP1 integrates an AMS CMV4000 [12] global shutter Complementary Metal-Oxide Semiconductor (CMOS) image sensor. Its main features include 4.2MP resolution with pixel size of 5.5µm, a high speed 180fps frame rate, selectable Analog-to-Digital Converter (ADC) resolution, 10 or 12-bit pixel width, image option Mono, RGB or NIR, output interface Low-Voltage Differential Signaling (LVDS) on 16 bits, up to 480Mbps. Data transfer is performed towards the OBC&DH FPGA via a serial SpW interface with a gross data rate of 128Mbps maximum.
In case of transmission of a full resolution 2048 rows per 2048 columns image, 12-bit pixel width encapsulated on a halfword (16 bits), transfer size is 64Mb. Transmission of reduced resolution images can also be performed if requested by Telecommand (TC), returning a 1MP image (1024x1024). In this case, time is shrunk by a factor 4. Supposing to perform a zero-overhead passthrough transfer to the on-board SDRAM buffer memory, this period represents the intrinsic latency of the subsystem for obtaining a full frame image from OP1. Data stored in SDRAM are thus available for readback by the dual-core LEON3FT SPARC V8 Central Processing Unit (CPU) aboard of the COBHAM Gaisler GR712RC [13]. This processor is installed on the OBC&DH aside of the FPGA. However, to access the SDRAM buffer the CPU must perform Memory-Mapped (MM) accesses to the addressable space of the FPGA. The latter is indeed the sole controller of the memory. These MM accesses are performed by means of a synchronous Input/Output (I/O) bus with 32-bit data width. System clock frequencies of CPU, FPGA and SDRAM buffer are set to 50MHz, thus leading to a potential 1.6Gbps gross throughput. This value is however reduced by a x25 factor circa due to access latency starting from the MMIO bus to the SDRAM memory, i.e. about 64Mbps overall (sustained performance). It is clear how speed data transfer requires to be fast for the target application. The high relative speed of the satellite wrt the asteroid would indeed lead to loss of target tracking in a short time. Figure 1 shows the FPGA architectural structure of the dataflow from OP1 to the MMIO bus. Also, considering the processing capabilities of the LEON3FT SPARC V8 CPU, the equivalent implementation of the algorithm would require 10s circa for a single image elaboration (see Section 5). These points represent the motivation for implementing the HW-accelerated image processing algorithm chain in the OBC&DH FPGA, previously indicated as IS. By elaborating the pixel data on-the-fly prior their storage in the SDRAM buffer the algorithm penalty time is assimilated. In such a way, the resulting image is already processed at its final storage time, ready for SW retrieval.

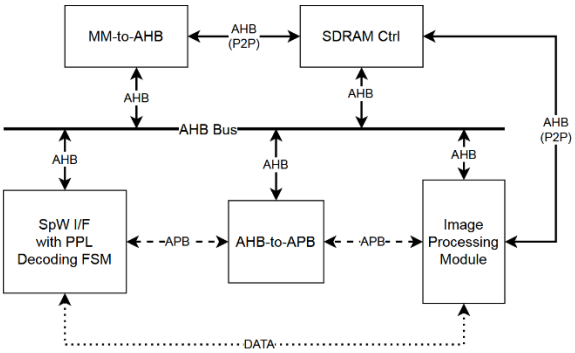

Figure 1: FPGA architectural structure

Image processing cores included in the IS datapath are followingly listed. The functional description of each one is to be subsequently found in the next separate paragraphs:

1. Pixel binning;
2. 2-D convolutional Low-Pass Filter;
3. Histogram Luminance generation with threshold calculation;
4. Binarization;
5. Multi-target Identification.

Pixel binning

The first functionality in the IS datapath is represented by pixel binning. This operation consists of an arithmetical average of blocks of 2x2 pixels, as shown in Figure 2.

$P_{(r, c)}$	$P_{(r, c+1)}$
$P_{(r+1, c)}$	$P_{(r+1, c+1)}$

Figure 2: Pixel binning scheme

In this scheme, r represents the row index and c represents the column of pixel P. This module therefore requires two image rows as input to generate in output a row of processed pixels of half the input dimension. Equation (1) represents the operation performed on each input block:

$$P'\left(\frac{r}{2},\frac{c}{2}\right) = \frac{P_{(r,c)}+P_{(r,c+1)}+P_{(r+1,c)}+P_{(r+1,c+1)}}{4} \qquad (1)$$

The VHDL implementation of this algorithm is shown in Figure 3.

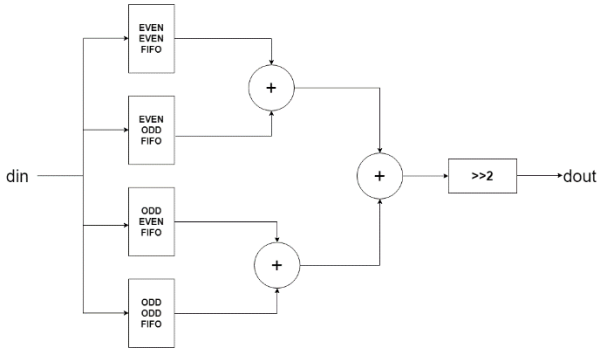

Figure 3: Binning datapath structure

This module can process both OP1 full and reduced resolution images. When operating on 2048x2048 (1024x1024) input images, the resulting binned image is therefore a 1024x1024 (512x512) processed data, with output data width equal to the input one. The core receives 1 pixel per clock cycle and provides 1 binned pixel every 2 cycles (1 for extracting data from FIFOs and summing two pixels intermediately, 1 for summing the two partial addendums and shifting the result right by two bits, i.e. dividing by 4).

Pixel binning is set as the first processing module to reduce image dimensions. Rationale of this choice stands on the LCC mission Deep Space scenario. Figure 4 shows a simulation of a picture acquired from OP1 during the closest-approach mission phase with the European Space Agency (ESA) Planet and Asteroid Natural Scene Generation Utility (PANGU) tool [14].

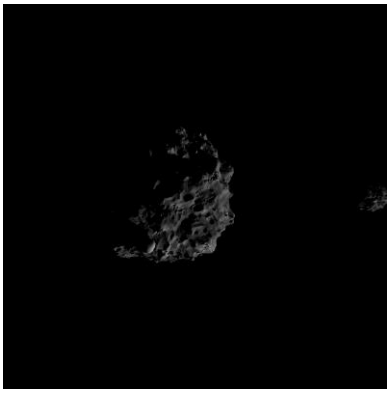

Figure 4: LCC OP1 image example

The Didymos asteroid is centered in the scene, with Dimorphos on the right-side edge. Autonomous tracking feature is based on recognizing non-black elements in the picture. By compressing the image size, the amount of pixel data to be processed is shrunk, thus reducing the time for target recognition. This elaboration is implemented by taking into consideration the assumption of the asteroid area in the image during the mission, still recognized after binning.

Low-pass filtering

A 2-D [15] Low-Pass Filter (LPF) is implemented as the following processing module. It performs a 2D convolution between the monochromatic input image and a 3x3 user-programmable kernel, individuated as follows:

$$\begin{bmatrix} K_{(0,0)} & K_{(0,1)} & K_{(0,2)} \\ K_{(1,0)} & K_{(1,1)} & K_{(1,2)} \\ K_{(2,0)} & K_{(2,1)} & K_{(2,2)} \end{bmatrix}$$

The core awaits input pixel data from the binning core of images with size 1024x1024 (512x512) and provides as output an image with same dimensions. The operation performed on each pixel $P_{(r,c)}$ is as shown in Equation (2):

$$P'_{(r,c)} = \frac{A_{r-1}+A_r+A_{r+1}}{9} \quad (2)$$

where A_i are the addendums shown as follows in Equation (3):

$$\begin{cases} A_{r-1} = P_{(r-1,c-1)} \cdot K_{(0,0)} + P_{(r-1,c)} \cdot K_{(0,1)} + P_{(r-1,c+1)} \cdot K_{(0,2)} \\ A_r = P_{(r,c-1)} \cdot K_{(1,0)} + P_{(r,c)} \cdot K_{(1,1)} + P_{(r,c+1)} \cdot K_{(1,2)} \\ A_{r+1} = P_{(r+1,c-1)} \cdot K_{(2,0)} + P_{(r+1,c)} \cdot K_{(2,1)} + P_{(r+1,c+1)} \cdot K_{(2,2)} \end{cases} \quad (3)$$

The graphic effect obtained by applying the LPF is smoothing the edges of an object in the picture, thus leveling out pixel values in the area. Moreover, background noise is filtered when applying the processing algorithm on a Deep Space image environment (i.e. one-pixel-dimensioned stars are removed at the output to avoid false target identification).

The arithmetical datapath is as shown in Table 1, where each multiplier M and adder A refer to Equation (3).

Table 1: LPF datapath pipelined structure

Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5
M0	A0	A4	A6	A7	*dout*
M1					
M2	A1				
M3					
M4	A2	A5			
M5					
M6	A3				
M7					
M8	A4	A4	A4		
0					
				DIV, 9	

The core receives 1 pixel per clock cycle, and provide 1 processed pixel every cycle after an initial latency of 6 cycle to fill the pipeline (1 for extracting data from BRAMs and multiplying the pixels with the kernel, 4 for summing the partial addendums in a binary tree fashion and 1 for dividing the sum to restore the dynamic).

Color depth compression

Downstream the LPF module, a Color Depth Compression (CDC) is performed on the resulting pixel data. This operation consists of a truncation of the Least Significant Bits (LSBs) to scale the color dynamic down from 12 bits to 8 bits. This resolution was considered enough for an adequate continuation of the image processing datapath. The operation is performed combinatorially, thus no additional latency is introduced in the datapath.
The LICIACube mission also foresees the acquisition of *Science mode* pictures that are not exploited for autonomous tracking. These images bypass the IS datapath and reach the SDRAM buffer as they are transmitted from OP1. In these circumstances, CDC is not applied, and the entire pixel dynamic is retained.

Luminance histogram and threshold computation

Byte-width processed pixels are forwarded to the next module in charge of counting the occurrence of each value in the data dynamic. Each new element increases the number of pixels in the image with that value. The result of this operation is an image Luminance Histogram (LH).
The core receives 1 pixel per clock cycle and provides the calculated luminance threshold (further described) 32-bit counter value. The resulting LH is provided at the end of the elaboration of the entire image.
An additional computation is executed once the LH is complete at the end of the image. Luminance counter values are incrementally summed until a comparison with a user-defined threshold is attained. This element represents the image Background Area (BA, expressed in number of pixels) that must be covered prior to stop the incremental sum of the luminance counter values. Once this threshold is reached, the module returns the last luminance counter value that was added to the sum. This element will be returned to the user interface as indication of the Binarization Threshold (BT) for the next acquired image.

Binarization

In parallel to the generation of the luminance histogram the output of the CDC execution is sent forward to the binarization module. The operation performed on each pixel $P_{(r,c)}$ is as shown in Equation (4):

$$P'_{(r,c)} = \begin{cases} 1, & P_{(r,c)} > T \\ 0, & P_{(r,c)} \leq T \end{cases} \quad (4)$$

where T is the user-defined BT. The core receives 1 pixel per clock cycle and provides the result combinatorially, thus no additional latency is introduced in the datapath.
The BT is the sole parameter defining either a pixel is to be considered black (0) or white (1). In the former case, pixel is leveled to background, thus ignored from following target recognition operations. In the latter, pixel dynamic is flattened to a single value to minimize the effort for object definition. The BT is available for programmability via a MM write operation on the AMBA APB [16] bus. This register is intended to be set prior the start of image transmission from OP1. As an example of interaction of the LH and threshold computation module with the binarization core, let the image previously depicted in Figure 4 be considered. If the BA parameter is set to 95%, the computed luminance value is 20. This number is read back by the user at the end of the elaboration of the current image. Once written back to the BT register, the binarized picture in Figure 5 is obtained.

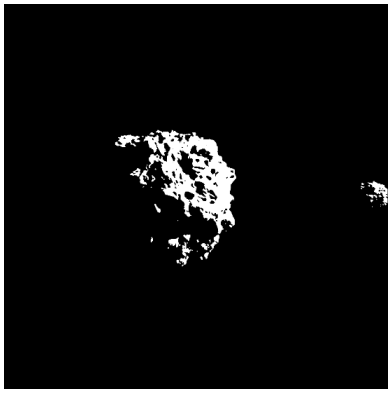

Figure 5: LCC OP1 image example after binarization

Multi-Target Identification

The last step prior forwarding of the processed data to the SDRAM buffer consists of the Multi-Target Identification (MTI) core. The conjunction of the outputs from this section provides the elements for image feature extraction to the SW running on the OBC&DH CPU.
MTI core awaits input pixels from the binarization module and operates on each incoming pixel. Void pixels (leveled to background with value 0) are ignored from target recognition. This statement calls for a meticulous selection for parameter BT. In fact, flattening a pixel to background causes a double effect on the MTI operations. It allows for filtering image noise if accurately chosen, although reducing the chance to successfully individuate a target in the picture, if set to a too high value.

All non-void pixels are recognized as a single object by analyzing the area surrounding each pixel in the stream. Each time a new object is identified, a new base color *B* summed to a delta color *ΔB* is assigned to current pixel value. This number is called *label* and it is appointed when a new object is identified in the process. Base and delta color are available for programmability via MM write operations on the APB bus. These registers are intended to be set prior the start of image transmission from OP1.
Input pixels to the core consists of binarized byte-wide data with value 0 (black) or 255 (white). MTI process produces 16-bit pure binary unsigned data in the range [0; 65535], where each target is *labelled* with a non-zero value. The core can process both 1024x1024 and 512x512 pictures. Elaboration for one pixel evolves on a Finite State Machine (FSM) as follows:

- Retrieve state: read from BRAMs the surrounding pixels data, holding them into temporary registers;
- Analyze state: compare current pixel value with registers;
- Update state: overwrite the current and surrounding pixels value with the selected label.

It is clear how MTI operations introduce a serious performance bottleneck in this worst case. To overcome this issue, an input synchronization FIFO is placed at the output of the binarization core. In such a way, MTI is performed guaranteeing image data coherency. Figure 6 shows the output image after the MTI process.

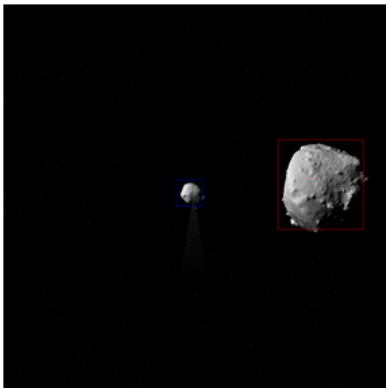

Figure 6: LCC OP1 image example after labelling

3. FPGA DESIGN RESULTS

Datapath and configuration

The image processing functionalities presented in the previous paragraph represent the modules composing the IS datapath, subsequently implemented targeting a Microsemi RT4G150-CG1657B FPGA device. The design was described in the VHDL language. Dataflow proceeds as follows:

- input pixels are decoded from OP1 SpW interface and forwarded to the binning module;
- a first stage of synchronization FIFOs is present prior the LPF core, which output latency is higher wrt its input one. CDC is automatically performed at the end of the LPF elaboration;
- resulting data is equally sent to the LH and binarization modules. Histogram is generated at the end of image transmission, along with threshold computation;
- a second stage of synchronization FIFO is present prior the MTI core, which output latency is higher wrt its input one;
- processed data from MTI are forwarded to a serializer module, in charge of packing the results for transmission on the 32-bit AMBA AHB [17] master interface, connected to the SDRAM controller;

An AHB arbiter manages the R/W requests on the bus, granting atomic access to the SDRAM buffer memory in case of multiple requests.

Control and configuration of the datapath is performed via a register bank accessed by the user via an APB slave interface. All I/O parameters are available for Read and Write (R/W) operations on this module, along with all cores status and statistics registers.

Device resource occupation

Table 2 lists the resources utilization when implementing the design on the targeted Microsemi RT4G150-CG1657B FPGA. Design tool is Libero SoC [18] v12.5 SP1 running on a Microsoft Windows 10 machine. Top-level includes all the non-listed elements of the design, i.e. synchronization FIFOs, APB slave and AHB master interfaces, register bank, serializer, DMA engine for accessing the SDRAM controller. The top-level overall utilization requires:

- 7928 out of the 151,824 available fabric 4-input-Look-Up-Tables (4LUTs), 5.2% circa;
- 4085/151,824 fabric D-type-Flip-Flops (DFFs), 2.7%;
- 36/210 Micro Static Random-Access Memories (uSRAM) of size 1Kbit, 17.1%;
- 30/209 Large SRAM (LSRAM) of size 18Kbit, 14.4%.

Table 2: IS resources utilization

Module	Fabric 4LUTs	Fabric DFFs	uSRAM 1Kbit	LSRAM 18Kbit
Overall	7928	4085	36	30
Binning	1200	564	8	0
LPF	1467	445	36	0
HL	627	362	0	3
Binarization	306	211	0	0
MTI	1673	718	0	12

The most relevant amount of resource occupation concerns the internal BRAM available in the FPGA device. This result is primarily due to the instantiation of the large synchronization FIFOs to guarantee data coherency while performing binning, LPF and OL operations. In fact, despite the module internal datapath pipelined structure allowing for concurrent R/W operations, these elements represent the performance bottleneck of the core. Their latency and the requirement to wait for entire rows prior starting the data elaboration do not allow to withstand the reduced performances of other units.

4. PERFORMANCES AND LIMITATIONS

Each processing module in the datapath can withstand a gross throughput of 50Mpixel/s at the OBC&DH FPGA nominal working frequency (50MHz), except for the MTI process, as followingly described.
Apart of this limitation, the bottleneck of this processing chain must be identified in the receiving SpW interface: due to the OP1 transfer limitations, the throughput of this interface is limited to 8Mpixel/s (i.e., 128Mbps). This statement is confirmed if the following assumption holds:

$$T_{SpW} < T_{min,IS}$$

where T_{SpW} T_{SpW} is the gross throughput of the SpW interface, while $T_{min,IS}$ $T_{\mathrm{min,IS}}$ is the throughput of the slowest modules composing the Image processing controller. Data moves forward in a streaming fashion as the image is received from the SpW interface. If any of the processing modules was to be slower than this interface, input data rate would be higher than output one, thus saturating the internal buffer data structures. In such a situation, data coherency would be lost along with results validity. Therefore, every receiving module must separately perform at a higher throughput than the transmitting side. Assumed the above potentialities of T_{SpW} maximally equal to 128Mbps and being known that each pixel is received on 16 bits, a minimum of 8 Mpixel/s must be processed by IS controller. As previously described, the highest latency among the modules is featured in the MTI process, requiring up to 5 clock pulses for processing a pixel after a complete row is received. Working with a nominal frequency of 50 MHz and with maximum 1024x1024 images, this means that maximum 5120 clock cycles are required to process a line (i.e., 102.4 µs). In the equivalent time, a gross amount of 1638 bytes are received from the SpW interface, thus requiring an at least equivalent FIFO to store the data not to overflow the buffer structures.

5. TESTING AND APPLICATION IN SPACE

Benefits of the FPGA-based datapath HW-acceleration are observed in a drastic processing execution time wrt SW run. As a reference, test execution results on Figure 4 will be presented.

Test setup

The test setup foresaw the utilization of an Electronic Ground Support Equipment (EGSE) composed of a workstation including a SpW PCI card, connected to the LCC OBC&DH Ground Model (GM) via a SpW cable. The PCI card was exploited as a OP1 emulator, sending the image data encapsulated on the correspondent protocol frame on the SpW interface with the target bitrate (128Mbps) as presented in Figure 7.

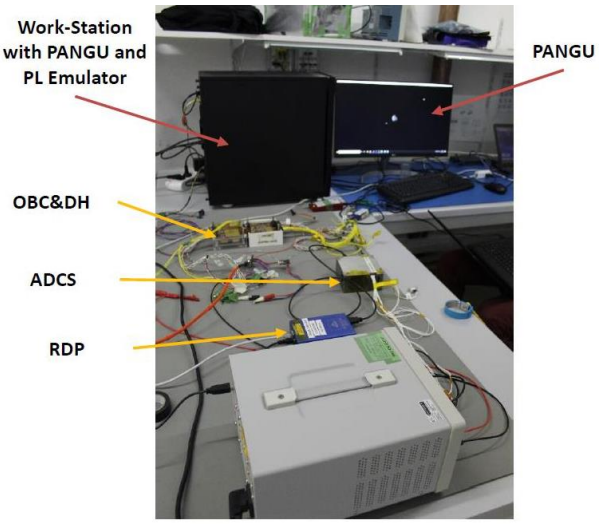

Figure 7: Test setup

Test results

To compare HW and SW performances, multiple runs were performed on the test image:

- IS-SW execution – the image received from the PCI card was stored in the SDRAM buffer memory by the FPGA prior being processed by the CPU after its retrieval. It consists of a pure C-language application, fully developed in-house, running on a single SPARC V8 CPU at 50MHz clock frequency, suitable to be run on

RTEMS. Measured time for elaboration started from the first pixel analyzed by the SW application;
- IS-HW execution – the image received from the PCI card was stored in the SDRAM buffer memory after being processed by the FPGA on the chain.

These outcomes are shown for each module in Table 3.

Table 3: HW vs. SW processing time comparison

Module	SW [ms]	HW [ms]
Binning	3446	653.93
LPF	4473	653.33
HL	505	653.36
Binarization	663	653.31
MTI	2186	653.56
Total	11273	653.93

Transmission of a full-size 2048x2048 12-bit image required 653.92ms in HW, including frame decoding and data storage. As the execution of the HW-accelerated image processing operations is performed in pipeline along with pixel reception, it can be noticed how the elapsed time is nearly transparent to this value. SW-to-HW total elapsed time ratio is about 17.2, showing an outstanding improvement of the overall image processing period.

Application in the LICIACube mission

The LCC satellite integrates autonomous navigation for close proximity fly-by of the Didymos binary asteroid system. Being the spacecraft relative impact speed with respect to the asteroid quantified to 6.5km/s at ≈50km closest approach, the Attitude Determination and Control System (ADCS) of the LCC satellite requires frequent correction inputs to keep the platform on track of the asteroids. Input data to this element are collected from the SW running on the satellite OBC&DH, which elaborates the images captured from OP1. The satellite will exploit the feedback results coming from the IS to recover the position and attitude of the satellite with respect to the target, collect data and apply a tracking strategy to keep the asteroid in the Field of View (FOV) of OP1 during the process.

The navigation system is composed of different modules:

- IS, receiving optical images of the target from OP1 and performing their processing;
- trajectory recognition module, which function is to reconstruct the orbit of the satellite from processed images to generate the desired attitude to perform the fly-by:
- attitude control module, that has the purpose of controlling the ADCS Reaction Wheels (RWs) in terms of angular speed, to generate the required torque vector for the fly-by.

The implementation of the IS oversees receiving images of the mission's target from the OP1 and performing image processing functions on the received data. It allows to recognize multiple objects in OP1 FOV and autonomously support attitude adjustment. Figure 8 shows the output feedback from the IS datapath.

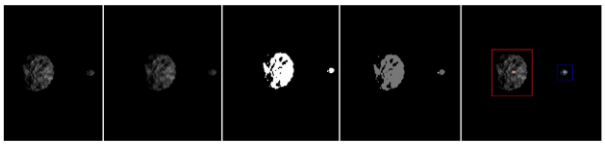

Figure 8: IS feedback output

Once the IS has provided its feedback, the tracking strategy can be implemented through the trajectory recognition module. Target tracking latency is therefore drastically shrunk, leading to a quicker control loop to operate on the satellite ADCS.

6. CONCLUSIONS

This paper presented the FPGA implementation of a HW-accelerated image processing chain to support high-speed object recognition leading to autonomous tracking. Starting from the definition of the required algorithms, an optimized solution in the VHDL language was provided, targeting a Microsemi RT4G150-CG1657B device. The proposed datapath integrates pixel binning, a 2-D convolutional low-pass filter, luminance histogram calculation, binarization, and multi-target identification. It allows flexibility with user configurability and easy integration with standard data interfaces. The module leads to high performances (up to 50Mpixel/s) and low latency, with pipelined elaboration and modest device resource occupation (5% LUT, 3% FF, 15% BRAM circa). With respect to a standard SW implementation, this solution can provide results about 20 times faster on a 2048x2048 12-bit grayscale image, thus allowing for highly reduced time for specific Space applications, such as per the LICIACube satellite autonomous tracking operations.

References

1. Lindoso, Almudena, and Luis Entrena. "Hardware architectures for image processing acceleration." Image Processing (2009).
2. Velten, Jorg, Marco Krips, and Anton Kummert. "Hardware optimized image processing algorithms for industrial high speed applications."

WSEAS Transactions on Computers 3.1 (2004): 1-6.

3. Comellini, Anthea, et al. "Vision-based navigation for autonomous space rendezvous with non-cooperative targets." 2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA. IEEE, 2020.
4. Li, Shuang, et al. "Image processing algorithms for deep-space autonomous optical navigation." The Journal of Navigation 66.4 (2013): 605-623.
5. Lentaris, George, et al. "High-performance embedded computing in space: Evaluation of platforms for vision-based navigation." Journal of Aerospace Information Systems 15.4 (2018): 178-192.
6. Mishra, Ashish, Mohit Agarwal, and Kota Solomon Raju. "Hardware and software performance of image processing applications on reconfigurable systems." 2015 Annual IEEE India Conference (INDICON). IEEE, 2015.
7. Gaias, Gabriella, Jean-Sébastien Ardaens, and Simone D'Amico. "The autonomous vision approach navigation and target identification (AVANTI) experiment: objectives and design." 9th International ESA Conference on Guidance, Navigation & Control Systems, Porto, Portugal. 2014.
8. Simonetti, Simone, et al. "LICIACube: technical solutions to monitor an asteroid space impact." EPSC-DPS Joint Meeting 2019. Vol. 2019. 2019.
9. Fazzoletto, Emilio, et al. "Autonomous Navigation Strategy on the LICIACube Micro-Satellite for Close Proximity Fly-By of the Didymoon Asteroid", 27th IAA Symposium on Small Satellite Missions, 2020
10. Argotec FERMI, Deep Space On-Board Computer 2020 Product Sheet, https://bit.ly/2QWvkEt
11. Microchip RTG4™ FPGA Datasheet, 2020, https://bit.ly/3fS9D0A
12. CMOSIS CMV4000 v3 Datasheet, 2014, https://bit.ly/3bZrIZH
13. COBHAM Gaisler GR712RC Dual-Core LEON3-FT SPARC V8 Processor 2018 Data Sheet, https://bit.ly/2ToriWf
14. ESA Planet and Asteroid Natural Scene Generation Utility (PANGU) Tool Enhancement, 2020, https://bit.ly/2SuUgDl
15. Di Carlo, Stefano, et al. "An area-efficient 2-D convolution implementation on FPGA for space applications." 2011 IEEE 6th international design and test workshop (IDT). IEEE, 2011.
16. AMBA 3 APB Protocol Specification, 2003, https://bit.ly/3us0ykz
17. AMBA 3 AHB-Lite Protocol Specification, 2006, https://bit.ly/3vmXKX4
18. Microsemi Libero® SoC Design Suite Brochure, 2017, https://bit.ly/3vqC4t3