EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESING (OBDP2021), 14-17 JUNE 2021

BOOSTING AUTONOMOUS NAVIGATION SOLUTION BASED ON DEEP LEARNING USING NEW RAD-TOL KINTEX ULTRASCALE FPGA

D. Gogu [(1)], F. Stancu [(1)], A. Pastor González [(2)], D. Fortún Sánchez [(2)], D. Gonzalez-Arjona [(2)], O. Müler [(3)], M. Barbelian [(3)], V. Pana [(3)]

[(1)] GMV Innovating Solutions S.R.L, SkyTower, 32nd floor 246C Calea Floreasca, Sector 1, 014476 Bucharest, Romania, and Email: info@GMV.es
[(2)]GMV Aerospace and Defence, Isaac Newton, 11 P.T.M., Tres Cantos, E-28760 Madrid, Spain, Email: info@GMV.es
[(3)] UPB-CCAS, Str. Gheorghe Polizu Nr.l, CP 011061, Sector 1, Bucharest, Romania, Email: octavian.grigore-muler@upb.ro

ABSTRACT

In this paper we present an ad-hoc architecture for on-board deep neural networks (DNN) inference implemented on a radiation-tolerant FPGA, creating building blocks which can be used for different DNN architectures, running on platforms with different amounts of resources and covering different requirements. The problem analyzed is based on an autonomous descent and landing scenario on the Moon, trying to compare it against traditional techniques. The implementation of the Deep Learning (DL) algorithm is focused on the extraction of features from navigation camera images. The proposed solution on FPGA allows for a reduced power consumption while maximizing the execution performance, as opposed to many on-ground solutions. A space-representative demonstrator has been developed to validate the solution.

Additionally, we present another approach to running inference of a DNN model on-board space representative avionics. In this case, it is based on using the specific Computer Vision and Artificial Intelligence (CVAI) accelerator Myriad 2 as the processing element instead of an FPGA. We describe the avionics architecture and the AI concept for two scenarios: crater localization on the Moon and specific patches detection on asteroid images.

1. Introduction

The majority of the embedded systems have been designed using linear algebra and linearization, but as the universe has non-linear behavior, they impose constraints and limitations on the potential current technology. This is true also for the space industry, where more demanding space missions could greatly benefit from the application of non-linear systems, specifically DL and DNNs.

The autonomous descent and landing on the lunar surface based on visual based systems is a complex challenge that can ensure pin-point landing. These Absolute Navigation systems are able to estimate a spacecraft position based on a previously generated landmark database and landmarks extracted from the navigation camera images using real time landmarks crater identification.

In this paper a more modern approach based on artificial intelligence (AI) solution is presented which benefits from the advantages of FPGA implementations.

The NN-VISNAV and AITAG are vision based navigation systems based on AI that can use common FPGA implementation. The NN-VISNAV is focused on descent and landing scenarios focused on the South Pole of the Moon. Two different landing points have been selected Schrödinger basin and Shackleton crater. The neural network is trained using different datasets of images which are generated in different conditions, like: illumination conditions, simulated image blur, navigation camera exposure and lens distortion.

In this paper we present two different approaches of how to apply inference of DNNs on on-board avionic platforms. This work stems from projects NN-VISNAV and AITAG developed in GMV for ESA.

We present the approaches taken in this two projects, where the inference of a DNN is executed on space-representative avionics. One of them uses an FPGA while the other uses an AI accelerator, the Myriad 2.

Two different visual-based scenarios are analyzed. Autonomous descent and landing on lunar surface and specific patches detection on asteroid observation. The FPGA-based approach covers the former while the AI accelerator covers both, while only partially the former.

We will firstly present the FPGA-based approach used in the NN-VISNAV project. A comparison between the conventional methods and the AI-based approach is

presented. The FPGA design is described in detail. Finally the validation is presented.

The AI accelerator-based approach is described with fewer depth, as the activity is still ongoing. The AI concept is described and the demonstrator avionics are presented afterwards.

2. Absolute navigation for descent and landing on the Moon surface

Visual-based autonomous descent and landing on lunar surface are a complex challenge. Systems capable of solving it are called Absolute Navigation systems, and can ensure pin point landing, estimating spacecraft position based on a previously generated landmark database.

The images are generated using planet and asteroid natural scene generation utility (PANGU), [11]. High definition models of the target surface have been developed using PANGU and are used to generate representative synthetic images.

2.1. Conventional vs AI-based approach

Previous work on using Absolute Navigation system conventional approach, presented in [1] and [2], showed a considerable increase in landing accuracy, from kilometers to a few hundreds of meters landing accuracy. The Absolute Navigation systems developed by GMV, in ESA – ANTARES and PILOT-B+, is based on recognition and matching of craters (as representative landmarks) on the lunars surface. Craters are excellent landmarks because can be found in large numbers with different size. The presence of different crates sizes has a big impact in the design of the AbsNav system which is targeting different craters' size at different altitudes, bigger craters at high altitude and smaller craters at low altitudes. Furthermore, the craters have rotation and illumination invariance properties which contribute substantially to the robustness of the Absolute Navigation system.

The conventional Absolute Navigation system is composed by two main parts: off-line part and on-line part. The off-line part is mainly focused on extraction of landmark databases which can be built in an automatic way using specially developed SW or manually using human interaction. In this process Digital Elevation Models (DEMs) or geo-referenced images can be used to extract the lunar landmarks to generate the database. The on-line part is mainly focused on on-board computation which ensures Real Time detection and matching of landmarks extracted from the images captured by navigation camera. During this process the craters are extracted from the images using optimized image processing algorithms followed by an estimation in shape and size using ellipse approximation. The last step is the matching procedure where the online extracted craters are matched with the ones existing in the landmark database. The matched set can be used to compute the absolute position in camera frame and a navigation filter can use the absolute navigation output together with on-board sensor measurement to provide a complete estimation of spacecraft's states. The conventional Absolute Navigation system approach is presented in *Figure 1*.

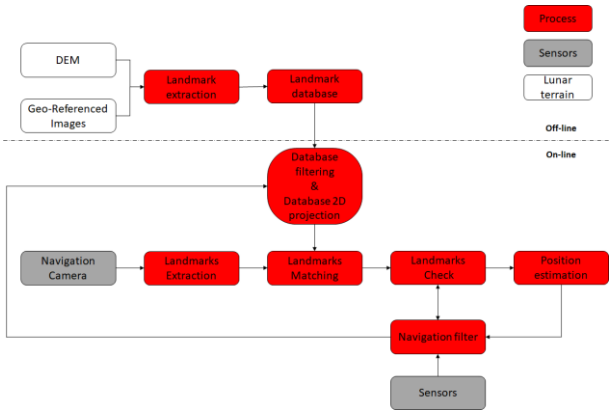

Figure 1: Conventional Absolute Navigation system architecture

The on-line part is composed by several functionalities designed to extract and match lunar craters in order to estimate spacecraft states: Landmarks Extraction, Database filtering & Database 2D projection, Landmark Matching, Landmark Check, Position estimation and Navigation filter. The Absolute Navigation system is working with 1024 pixel squared image. The Landmark Extraction is using the navigation image to extract craters and is composed by three stages: border detection where edge detection and crater rim detection is selected, rim grouping where crater curvature check and crater border coupling is performed and ellipse fitting where the size and shape of the crater is approximated.

The Database filtering & Database 2D projection is using the on-board Landmark database and estimated spacecraft states to select from the database the craters which are predicted to be seen in the camera FoV. Afterwards, the selected craters are processed by Database 2D projection algorithm which projects the craters in the image frame based on spacecraft's estimated attitude and position.

The Landmark matching functionality is performing the point matching by using the craters extracted from the image (Landmarks extraction) and the projected craters in image frame from the database (Database 2D projection). The output of Landmark matching is a list of crater pairs (the craters extracted from the image and its matches from the database) which are matched

during this procedure. The Landmark check is in charge to eliminate false matches based on proximity criteria (if near the matched pair exists other craters, relatively close to this pair, the match is discarded). The Position estimation is using attitude information knowledge, landmark database and matched craters to compute the 3D absolute position estimation. The navigation filter is a Linear Kalman Filter that integrates different sensors measurements in order to obtain precise estimated spacecraft's states.

In ESA project, NN-VISNAV, a new modern technique based on AI is implemented to perform visual navigation. The NN-VISNAV project is developed in collaboration with UPB-CCAS and implies to provide a DNN complete solution, starting with research and SW implementation until HW implementation on space-graded FPGAs. The preliminary evaluation of neural network solution and implementation is performed in TensorFlow and neural network training is performed using 2048 pixel squared image. The HW implementation of the neural network is performed by GMV.

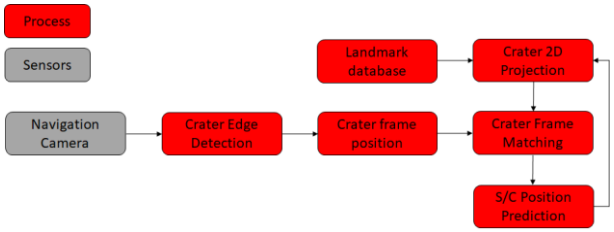

Figure 2: AI-based Absolute Navigation architecture

The Crater Edge Detection is designed using neural network techniques, trained with perturbed and non-perturbed data (in this case images) with the scope to perform crater edge detection. The crater frame position is a SW function that identifies and approximate craters shape and position in the image by using the extracted edges. The crater 2D projection is in charge to select and project crates from database in the image frame by using the spacecraft estimates states, similar as in the classical Absolute Navigation system. Then Crater Frame Matching is in charge to perform the matching of the craters detected in image frame with the ones from crater database. The Spacecraft Position Prediction function is in charge to estimate spacecraft estimated states. The presented approach for AI based Absolute navigation assume a HW-SW co-design approach, where Crater Edge Detection neural network is implemented on dedicated FPGA to increase the execution performances and the rest of the functionalities are implemented on a space graded processor.

2.2. DNN inference on FPGA

The avionics for this approach is formed of the main OBC, where images from an on-board optical device are collected. These images are then to be subjected to inference using a DNN model, previously trained offline. The execution of the inference processing is carried out in an FPGA, leveraging its capabilities to parallelize multiple processing operations and its higher power efficiency with respect to other solutions such as CPUs or GPUs. In this way, the OBC acts as the client which requests the execution of the inference of an input image to the FPGA, which acts as the server.

The interface between the OBC and the FPGA is built on an Ethernet connection, using a tailored protocol working over raw Ethernet packets, i.e. the network, transport and application layers are custom. The designed protocol supports for packet fragmentation, and multi-packet acknowledgment. The application layer is based on a series of commands that enable the client to send images, request execution of inference, retrieve results from intermediate and final layers, and monitor the status of the server.

In order to generalize to different DNN architectures, specific implementations for each type of layer and its parameters are described in a Hardware Description Language (HDL) and verified. Through parameterization these modules can be programmatically generated and generic models be constructed, as long as they are built up from implemented layer types. The sequence of execution steps and control signals can be generated offline and then be fed to a controller module that internally handles the orderly execution of each instruction.

From a model generated using TensorFlow and Keras, a Python library has been generated in order to extract the required information: layer types, input and output dimensions, weights and biases. This library then generates the required inputs for the FPGA implementation: scheduling instructions, parameter files, and memory distribution.

A demonstrator of this architecture has been built under the NN-VISNAV. The used FPGA board has been the Alpha Data SDEV Kit-2, which contains a Xilinx Kintex UltraScale KCU060 FPGA, of the same family of the radiation tolerant XQRKU060 FPGA. An Ethernet expansion module was used to connect the board to a workstation, which is used to emulate the OBC and the camera. The setup for the testing of the demonstrator is shown in *Figure 3:*.

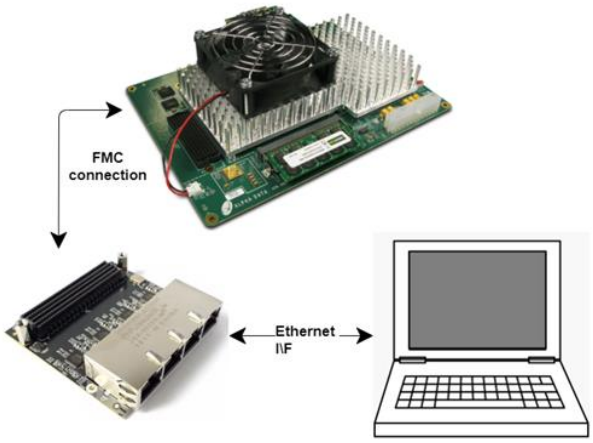

Figure 3: Setup of the demonstrator

In order to ease its use and to abstract the complexity of the inner layers, a Python-based command line tool has been developed to control the execution of the different tests on the demonstrator. A series of commands can be used to carry out all the steps necessary to prove the functionalities of the system. This includes loading the bitstream, loading model parameters, sending query images, executing the inference, retrieving results from any layer of the model, and monitoring the status of the system.

The U-Net model implemented in the demonstrator implements the crater edge detection step of the Absolute Navigation. It expects a 2048x2048 8-bit pixels input image, and generates a 2048x2048 pixels output feature map. It contains 30 layers of different types (Conv2D, DepthwiseConv2D, MaxPool2D, Concatenation) and with different activations (ReLu and sigmoid).

2.3. FPGA design

Our design is based on a library of in-house VHDL modules, which allow for a modular implementation of generic DNN architectures.

This modular solution eases the adaptation to different FPGA platforms with different amounts of resources, as it allows for trade-offs in the utilization of specific resources such as BRAMs and DSPs, where the bottleneck could vary from platform to platform.

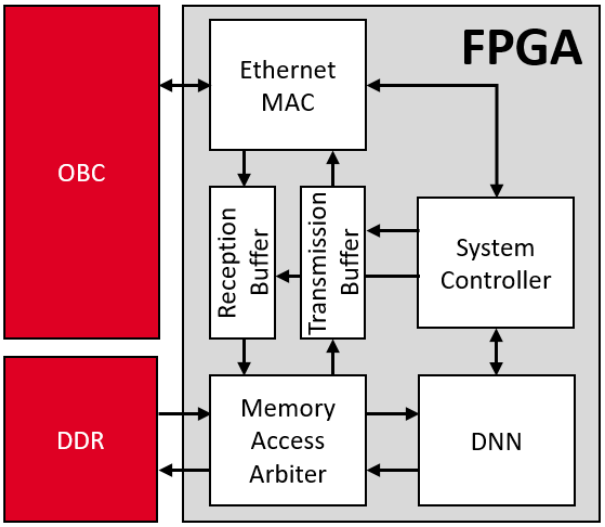

Figure 4: Avionics diagram

The communication with the client is handled by an Ethernet MAC implementation and a system controller. The system controller continuously monitors the status of the interface, and manages both the retrieving of incoming packages and the transmission of outbound packets. The full application layer is also managed by the system controller, which translates incoming commands from clients into specific actions within the design and generates and transmits the proper response back. This involves, among others, the control of the data flow between the DDR and the outside world. It manages the fragmentation of transmission packets and the addressing and redirection of reception packets. It also takes care of the acknowledge mechanism by which multiple packets can be sent in a burst and an acknowledge for each of them is expected from the receiving side, otherwise having to re-transmit the unacknowledged packets.

2.3.1. DNN implementation

The architecture of the DNN is formed of two parts. The controller and the processing pipeline. The controller manages the accesses to memory, the sequence of steps necessary for the execution and the interface with the outside world. The processing pipelines contain the main processing elements, the Processing Units (PU), which are the elements that effectively execute the arithmetic operations that constitute each DNN layer.

The architecture is configurable and adaptable to the available resources as well as to the characteristics of the DNN, enabling a trade-off between resources, performance, and power consumption.

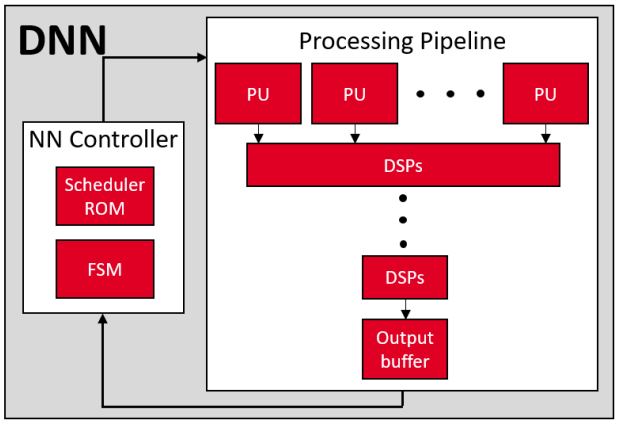

Figure 5: Block diagram of the DNN

The control of the DNN is exercised by several Finite State Machines (FSM). In this way, the system handles accesses to memory, in order to read parameters and inputs and to write outputs of each layer. It is also responsible for activating the PUs required for each layer, and acts as the interface for controlling and monitoring the status of the DNN implementation from the outside.

The controller has a scheduler ROM, whose content is loaded in the initialization phase with the required instruction sequence to control the execution of the inference. This way, the instructions are fetched as required, traversing the sequence of commands once for every input image.

The memory distribution made is designed so that there is no data overlap in memory. Each layer of the DNN has an allocated memory block in which to store the output tensors. Subsequent layers may access blocks of their previous layers, and this process is repeated sequentially until all layers are traversed.

The DNN parameters have their own allocated memory block, and are loaded into memory in the initialization phase. The used memory layout for weights is CoutCinHW, and CHW for intermediate feature maps, where C is the channel dimension, W the width, H the height, and Cin and Cout the input and output channel dimension of weights. Biases are interleaved with the weights around the Cout dimension.

The PUs in charge of performing the processing are composed of the following elements: an Upsampling block, a MaxPooling block, Convolution blocks and BRAM memory blocks that act as input caches. A PU can only perform one type of operation at a time, although in the case of convolutions it can simultaneously use multiple blocks on smaller input images.

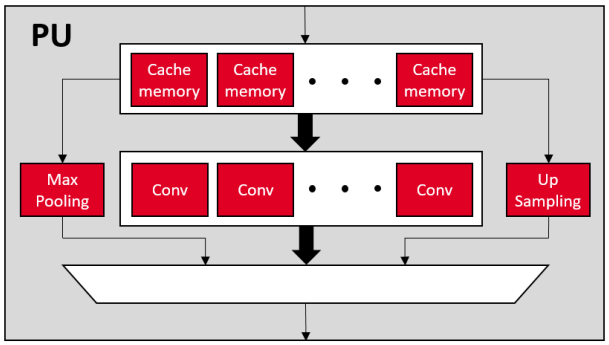

Figure 6: Block diagram of the Processing Unit

As convolution requires in general multiple input pixels per clock cycle, a large number of memory blocks are needed in the PU. The size required for these memories is determined by the maximum size of the channels and can be adjusted depending on the DNN to be implemented. Similarly the number of PUs determines the number of channels that can be calculated simultaneously and can also be adjusted if the DNN to be implemented is different.

Inside the DNN but external to the PUs there are multiple DSPs, placed by levels in an invert pyramid, which add the results of the convolutions and the corresponding bias until obtaining the result. This value may then be optionally passed to an activation function (ReLu or sigmoid). For the Xilinx UltraScale FPGA family, these DSPs are capable of performing multiplications of up to 27x18 bits, giving an output of 48 bits.

The internal representation of values in the FPGA is 16 bits fixed point. The number of integer and decimal positions to be used needs to be calculated using representative images with the trained model. In order to limit the precision loss do to the use of fixed point arithmetic, the whole set of observed values is inspected. The most significant bit required to represent the values observed in each layer determines the 16 bits that are used in the FPGA.

2.4. AI HW validation

2.4.1. NN-VISNAV Validation

The DNN for visual based navigation is implemented in VHDL which involves transformation in fixed point arithmetic. During fixed point arithmetic transformation process the neural network is transformed from 32 bits floating point to 16 bits fixed point, which can cause losses in edge detection accuracy. An extensive evaluation validation of the neural network VHDL implementation is performed by evaluating the response at edge level until final results of the absolute navigation system. The validation of the HW implementation results is performed with respect to SW implementation results. The evaluation is performed

using the nominal scenario which involves descent and landing in Shackleton crater. For validation purposes the SW results of the neural network are considered to be the reference.

The Crater Edge Detection neural network provides a mask of the crater edges that are present in the image The HW implementation results are evaluated at pixel level by comparing the edges in both SW and HW implementations. They are proved to match for 98.6% of the pixels, which ensures high degree of result similarity. The crater edges extracted using the HW neural network implementation are used to run the Crater Frame Position and the Crater Frame Position in order to evaluate the overall results of the AI based Absolute navigation system. The estimated center and radius of the craters error with respect to SW results were found to be subpixel.

Table 2.4.1-1

	Center detection precision on x axis	Center detection precision on y axis	Radius detection precision
Mean	0.83	0.75	0.29

The number of common matches is an important metric to be evaluate in order to fully validate the HW implementation. Similar number of matches are seen over the entire nominal scenario execution:

Table 2.4.1-2

	SW Matches	HW Matches
Mean	171.74	171.41

The following graph illustrate the comparison between the crater matches of the SW implementation vs HW implementation. As can be observed the output between both implementation is very similar.

Figure 7: Number of matched craters validation, SW vs HW

The output of the Absolute Navigation system based on the DNN HW implementation is presented in *Figure 8*.

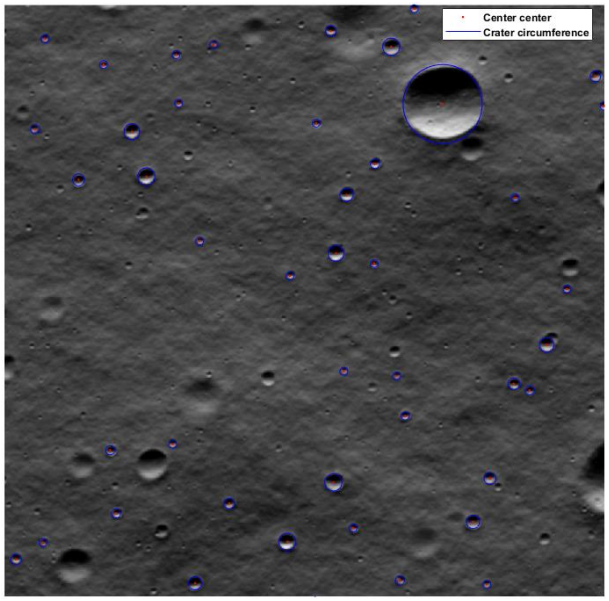

Figure 8: AI based Absolute Navigation HW results, top left corner zoomed area

In *Figure 8* the red dot represents the crater estimated center and the blue circle is the size of the crater using the estimated radius.

2.4.1.1. Crater matching and localization on Moon surface

The objective of this scenario is to use DNNs to be able to identify and locate craters in visual images of the Moon in descent and landing trajectories. This approach aims to produce a list of detected craters as well as their location and apparent size using a single DNN. This may result in an improvement over other approaches that make use of conventional methods or even other AI based systems that limit their DNN to predicting a mask of crater rims. If this approach produces adequate performance results, it could well substitute many iterative steps that were conventionally executed on SW

2.4.2. AITAG Validation

Training DNNs requires extensive amounts of images. Applicable space images are scarce and their labelling is costly, so for the AITAG scenarios, training will be done using synthetic images. Available real images will be used in the validation phase however, so once finished this activity should provide results on how well the training on synthetic images ports to real imagery.

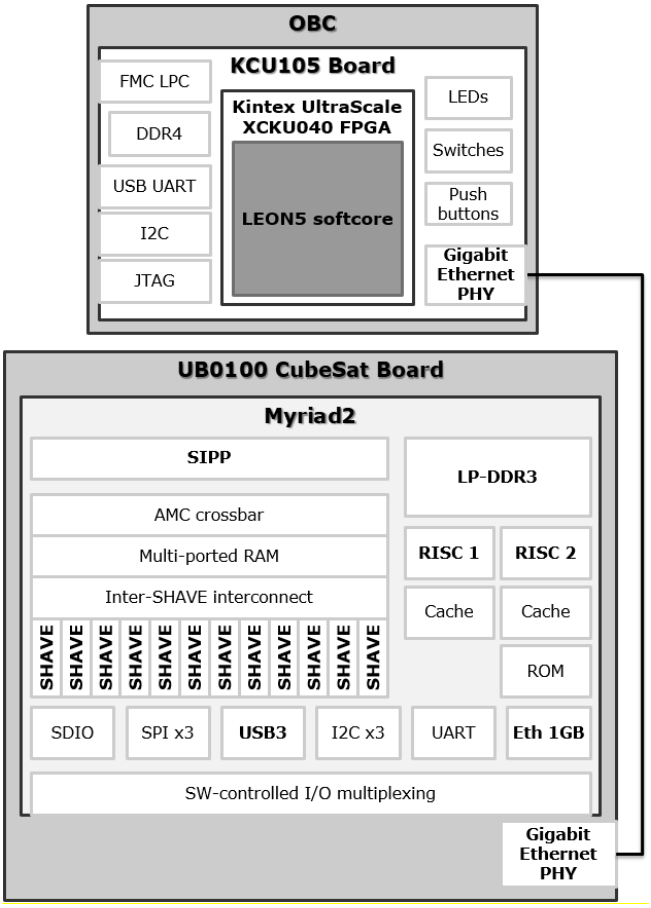

Figure 9: Avionics architecture using the Myriad 2 CVAI accelerator

2.4.2.1. DNN inference on AI accelerator

The second approach to executing inference of DNNs on on-board avionics is the use of specific AI accelerators. The avionics in this case are formed of the accelerator component that implements the DNN, and the OBC which is in charge of configuring the accelerator, feeding the images, and retrieving the results.

In the AITAG activity, a demonstrator is being built in order to test this approach. The selected accelerator is the UB0100 CubeSat Board from Ubotica. It is based around the Intel Movidius Myriad 2 Vision Processing Unit (VPU). They are designed for application to image processing and NN inference. This VPU is designed for use in low-power edge applications providing in excess of 1 TOPs of compute power. This VPUs have also been characterized for their use in space.

The OBC is implemented on a Xilinx KCU105 board, which contains a Kintex UltraScale KCU040. This FPGA is based on the same architecture as the radiation tolerant XQRKU060. A LEON 5 softcore microprocessor is implemented on the FPGA fabric and is the responsible of managing the interface to the external world, configuring and monitoring the accelerator, and providing the images as well as retrieving the results.

2.4.2.2. Specific patches detection on asteroid images

The goal of the second scenario analyzed in the AITAG project, the asteroid scenario, is to be able to detect selected visual patches on images of an asteroid. This scenario is based on the HERA spacecraft of the ESA-NASA AIDA mission. The spacecraft will measure the impact and the asteroid deflection on the Didymos double asteroid made by the DART impactor. It will enable characterization of the volume and surface properties with different instruments, including a camera, done on successive hyperbolic trajectories.

On the first stages of the observation, camera images are sent to ground, where scientists can decide which patches of the surface have more scientific interest. The goal of the scenario is to be able to detect in real time when one of these patches of interest appears in the images from the camera. For that purpose a DNN-based system has been designed and is currently being developed. This DNN system would enable the autonomous orientation of the camera towards these patches, and it would also reduce the required downlink throughput for camera images, as it would enable to filter the images based on their content.

A conventional approach to tackle this problem would require characterizing one or multiple specific features of the patch, some kind of descriptor, and then detecting this same features on the query images. The proposed DNN design relies on a Siamese network to compare and measure the similarity of the reference patch and the query image from the camera. Two parallel base architectures generate a feature map for each the reference patch and the query image, which acts as a descriptor. A distance function between descriptors is used to obtain a similarity metric between them.

The system is trained with a triplet loss function, which aims to reduce the distance between equivalent patches and to increase the distance between unrelated patches. In this way, the system generates a similarity matrix for each image where a threshold value may be applied to the maximum similarity values to detect the reference patch.

3. Conclusions

This paper presented a modern AI based Absolute Navigation in a HW-SW co-design where the most computation intensive processing is implemented in a dedicated FPGA. The performance is boosted by taking advantage of the characteristic data flow pipelining and parallelization characteristic of FPGAs. The edge detection is the best candidate taking into consideration that large images of 2048x2048 pixels are used to extract navigation information.

Processing Units are the elements that effectively execute the arithmetic operations that constitute each DNN. By adjusting the amount and size of its components, PUs maximize the flexibility of the design for future implementations, enabling tradeoffs between utilization of resources and processing speed. Different DNNs can be more easily adjusted to fit in different FPGA platforms and to optimize the utilization of the resources available.

The DNN from the NN-VISNAV scenario is fully implemented in VHDL and contains 30 layers. This involves the utilization of approximately 96% of the critical component, the BRAMs.

The HW implementation of a DNN offer subpixel errors on center and radius estimation with respect to the SW implementation. Also the output of common matches is similar over the entire nominal scenario execution.

ACKNOWLEDGEMENTS

This research effort was sponsored by the European Space Agency (ESA) contract number 4000122394/17/NL/Cbi/hh (NN-VISNAV) and 4000130092/20/NL/CRS (AITAG). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the European Space Agency.

REFERENCES

[1] M. A. Rodrigálvarez, M. Mammarella, A. M. Sánchez Montero, Pablo Colmenarejo, 2011, System Design and Performance Evaluation of Advanced Optical Terrain Absolute Navigation for Pinpoint Lunar Landing, 8th International ESA Conference on Guidance, Navigation & Control Systems.

[2] M. Mammarella, M. A. Rodrigálvarez, A. M. Sánchez Montero, Bach Van Pham, Simon Lacroix, 2010, Comparison of Optical Terrain Absolute Navigation Techniques for Pinpoint Lunar Landing, IAC-11-A3.2B.7

[3] Sze Vivienne, Chen Yu-Hsin, Yang Tien-Ju, Emer Joel S., Efficient Processing of Deep Neural Networks: A Tutorial and Survey, Proceedings of the IEEE, Vol. 105, Issue: 12, pp. 2295-2329, Dec. 2017

[4] David Gonzalez-Arjona, Lorenzo Cercós Pita, Adrian Danciu, Marcos Avilés Rodrigálvarez, Klaus Hornbostel, Marco Mammarella, Imanol Cruz, Cristian Corneliu Chitu, Exploration Symposium (A3), Advances in the Hardware/Software co-design for the Absolute and Relative Vision Based Navigation systems for the Lunar Landing Scenario, 66th International Astronautical Congress 2015

[5] R. Szeliski, S.B. Kang, Recovering 3-D shape and motion from image streams using non-linear least squares, Journal Visual Communication and Image Representation, vol. 5, No.1, pp 10-28, March 1994

[6] Larry H.Matthies, Andrew E.Johnson, Precise Image-Based Motion Estimation for Autonomous Small Body Exploration, Proceedings of 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (ESA SP- 440), pp.627-634, ESTEC, Noordwijk, 1-3 June 1999.

[7] Gupta, P., Loparo, K.A., Mackall, D., Schumann, J., and Soares, F.R, "Verification and Validation Methodology of Real-time Adaptive Neural Networks for Aerospace Applications", February 2014.

[8] Manning, J., Langerman, D., Ramesh, B., Gretok, E., Wilson, C., George, A., MacKinnon, J., and Crum, G, "Machine-Learning Space Applications on SmallSat Platforms with TensorFlow", November 2019.

[9] "Deep Learning Crater Detection for Lunar Terrain Relative Navigation", Downes, L., Steiner, T. J., and How, J. P, January 2020.

[10] "U-Net: Convolutional Networks for Biomedical Image Segmentation", Ronneberger, O., Fischer, P., and Brox, T, May 2015.

[11] Planet and Asteroid Natural scene Generation Utility, University of Dundee, UK, https://pangu.software/