The SPARTAN Project: FPGA-based Implementation of Computer Vision Algorithms Targeting to Space Applications

D. Diamantopoulos, G. Lentaris, A. Douklias, K. Siozios, D. Soudris
School of Electrical and Computer Engineering
National Technical Univ. of Athens, Athens, Greece
{diamandt, glentaris, douklias, kpsiop, dsoudris}@microlab.ntua.gr

M. Avilés Rodrigálvarez
Advanced Space Systems and Technologies
GMV
Madrid, Spain
maaviles@gmv.com

Abstract— Vision-based robotics applications have been widely studied in the last years. However, up to now solutions that have been proposed were affecting mostly software level. The SPARTAN project focuses in the tight and optimal implementation of computer vision algorithms targeting to rover navigation for space applications. For evaluation purposes, these algorithms will be implemented with a co-design methodology onto a Virtex-6 FPGA device.

Keywords: Computer Vision Algorithms, Space Application; FPGA implementation; High-Performance; HW/SW co-design

I. INTRODUCTION

The exploration of Mars is one of the main goals for both NASA and ESA, as confirmed by past and recent activities. The last 15 years there are a number of on-orbit and surface missions to Mars with remarkable results. The impressive success of previous missions (NASA’s Mars Global Surveyor, Mars Odyssey, Phoenix, Mars Reconnaissance Orbiter, as well as ESA’s Mars Express) enables a continued investment of efforts in this direction, and also facilitating international discussion for joint Mars exploration missions between ESA and NASA, such as in case of ExoMars which is scheduled for 2018 [1].

However, in order to accomplish this goal, a number of critical issues have to be resolved. Among others, efficient autonomous robots have to be designed. Autonomous robots’ behavior greatly depends on the accuracy of their decision-making algorithms. Hence, vision-based solutions are becoming more and more attractive due to their decreasing cost, as well as their inherent coherence with human imposed mechanisms. In the case of stereo vision-based navigation, both the accuracy and the refresh rate of the computed disparity maps are the cornerstone of success. However, robotic applications place strict requirements on the demanded speed and accuracy of vision depth-computing algorithms.

On supporting this goal, at this project we aim to provide a sufficient implementation of computer vision algorithms for rover navigation. Rather than similar approaches that solve this problem solely in software-level, our objective is to provide a number of computer vision algorithms in ESA compatible VHDL format [4], whereas the implementation of SPARTAN system will be performed using a co-design methodology, a ROS-based system [5], and a Virtex-6 FPGA device[2].

The rest of the paper is organized as follows: Section II describes the SPARTAN system architecture and the employed computer vision algorithm, whereas section III presents a novel co-design methodology for computer vision algorithms. Section IV discusses the implementation of SPARTAN system. Finally, section V concludes the paper.

II. SPARTAN SYSTEM ARCHITECTURE

The goal of the SPARTAN processing algorithms is to convert visual information from the rover cameras into 3D local maps and perform accurate location estimates useful for the navigation process. Image processing algorithms suitable for localization and 3D map reconstruction are selected and implemented into a parallel processing chain to achieve high performance while maintaining efficiency in terms of energy use, computing power and memory footprint.

The CV algorithms will be partitioned into a SPARTAN processing software component and a VHDL model component. The former will run on a standard central processing unit with the GNU/Linux operating system, whereas the latter will be implemented in a recent FPGA device. The system will use stereo-vision image information in order to feed SPARTAN algorithms. From a systems’ perspective, SPARTAN algorithms need to interact with other components of the rover navigation application for them to be useful. Interfacing between the rover sensors and the processing elements will be accomplished through central processing component.

The main objective of our solutions is to achieve the desired rover navigation function, while reducing as much as possible the required overall budgets. For this purpose, a number of novel computer vision algorithms will be developed both in C/C++, as well as VHDL language.

A schematic view of the proposed solution is depicted in Figure 1. More specifically, the computer vision algorithms that will be implemented during the SPARTAN project are summarized as follows:
- Imaging: Implementing suitable local image processing that can serve image products.
- Visual Odometry: Provide an estimation of the displacement of the rover.
- Visual SLAM: Determine the current location of the rover.
- 3D Map reconstruction: Reconstruct the 3D-dimensional shape of the terrain being imaged in front of the rover.
- Localization: Locates the new spatial location of rover at the map.

Figure 1. SPARTAN functions identification

Since our goal is to achieve an efficient implementation of these algorithms, special care should be applied both during the selection of them, as well as at their development. Furthermore, in order to achieve additional performance improvement, a number of source-to-source modifications might be applied.

III. THE EMPLOYED CO-DESIGN METHODOLOGY

The proposed co-design methodology is depicted in Fig. 2. As input we use a high-level description of SPARTAN architecture (e.g. in C/C++). This methodology will be applied to the entire SPARTAN system in order to clarify either which building, or the distinct functionalities of each building block, is candidate for implementation onto the FPGA platform. The flow is completed with the system integration and back end design phases. During system integration two different evaluation procedures will take place. Initially, each of the developed kernels will be validated in order to verify its proper functionality in respect to timing, performance, and throughput constraints posed by the specifications of computer vision algorithms. Possible violations to these constraints will be alleviated by applying focused optimization techniques (e.g. through usage of pipeline/parallelization techniques). After this task, we can guarantee that all the developed kernels meet system’s specifications. Then, one more evaluation step takes place, where the entire system is evaluated against to the same criteria (proper functionality in respect to timing, performance, and throughput constraints). Potential violations in this task will be overcome through appropriately tuning of kernels where validations are identified. while it consists of the following steps: profiling, behavioral optimization, partitioning, kernels mapping into FPGA, kernels mapping into CPU, system integration and back-end tools and configuration and run-time execution.

Thus, the proposed VHDL modeling methodology consists of the following steps: profiling, behavioral optimization, partitioning, kernels mapping into FPGA, kernels mapping into CPU, system integration and back-end tools and configuration and run-time execution.

This methodology provides a number of trade-offs that balance various design criteria among the FPGA device, the employed CV algorithms, as well as the entire SPARTAN system. More specifically, these trades-off parameters are summarized as follows: complexity reduction, performance improvement, memory management, parallelism and pipeline extraction.

The outcome from profiling analysis provides some guidelines, which is valuable information for the HW/SW partitioning. More specifically, the conclusions derived from profiling procedure are to identify kernels of CV algorithms that are:
- Suitable for implementation onto HW: These kernels usually contain arithmetic functions which mostly involve matrix, or vector, multiplications. These functions are candidate for being mapped onto HW, rather than executing onto a general-purpose CPU, because they could be parallelized. Additional performance enhancement could be derived by employing pipelining techniques.
- Suitable for implementation onto SW: The kernels that are suitable to be mapped onto SW include mostly functionalities that belong to control path, which is very
difficult to be implemented onto HW. Additionally, since these functions include mostly code with dependencies (serial execution), it is not possible to parallelize.

- Need additional study in order to derive the target implementation platform: Finally, there are some kernels of the employed CV algorithms that need further investigation in order to derive a conclusion about their target implementation platform (HW or SW).

In order to perform the previous analysis we extract the following metrics from our profiling methodology:

a) The application’s task graph, which gives the functional dependency among kernels of the SPARTAN system.

b) The data flow that depicts how data (e.g. arrays) are propagated among kernels of the SPARTAN system. This information is also valuable in order to identify which of these kernels are candidate for being executed in parallel.

c) The memory footprint in order to determine the maximum memory requirements for the entire SPARTAN system.

d) The data lifetime that denotes how the memory requirements are distributed to the time.

e) The relative computational cost, which is expressed in terms of mathematical operations executed in each kernel of CV algorithms.

IV. SYSTEM IMPLEMENTATION

Since the goal of this project is to provide efficient computer vision algorithms with increased performance, our methodology identifies those functionalities that introduce performance bottleneck in systems’ execution and handled appropriately through source-to-source modifications. These modifications will take into account inherent constraints/features taken by the target platform.

The HDL descriptions of the target system’s modules will be mapped on reconfigurable hardware blocks of appropriate type using systematic approaches, mapping tools and reconfigurable hardware generators that are commercially available (Xilinx EDK Framework [3]). At this point and taking into consideration the partitioning decisions, an appropriate interconnect network will be produced using the Xilinx EDK Framework. The mapping procedure assumes the existence of the FPGA and CPU hardware modules, as well as the hardware infrastructure of the interconnect network.

A. PC-FPGA Interfacing

Regarding the interface between host PC and the FPGA device we define two communication stages, the configuration stage and the debugging and execution stage. Figure 3 summarizes the communication protocols for both stages.

The selected interface for FPGA to Central processing (Linux board) communications at the programming stage is the JTAG interface. FPGAs are JTAG-aware and so all the FPGA I/O pins can be controlled from the JTAG interface. FPGAs add the ability to be configured through JTAG (using proprietary JTAG commands). JTAG allows device programmer (hardware) to transfer data into internal non-volatile device memory. In the case of FPGAs, volatile memory devices can also be programmed via the JTAG port normally during development work. The Xilinx ISE tools can then be used to read-back the FPGA configuration for example or to even inspect internal signals, using “ChipScope Pro”.

![Communication protocols](image)

The SPARTAN system shall use extensively communication scenarios between the host PC and the FPGA, during both the design and execution stage. Depending on amounts of data which should be transferred, different types of connections can be used. In order to determine those connections and provide better communication flexibility, we define the system architecture in terms of logic layers. The system layers, as shown in Figure 4, are: FPGA Hardware layer, FPGA Software Stack layer, Channel layer, Host PC Software Stack layer and Vision Algorithms Wrappers layer.

- Vision Algorithms Wrappers Layer: Contains the control section of computer vision algorithms of SPARTAN system written in C/C++. All requests to hardware are initiated by those programs.

- Host PC Software Stack Layer: It is responsible for translating the requests from Vision Algorithms Wrappers Layer to communication protocol oriented requests.

- Channel Layer: It is responsible for accessing hardware using selected physical interface.

- FPGA Software Stack Layer: This layer contains the appropriate libraries that support the communication protocol solution of Channel layer. These libraries may be simple RS232 software interfaces up to complete Linux kernel images containing i.e. the Ethernet software stack.

- FPGA Hardware Layer: Contains the architecture of vectorial processing. In this layer except of hardware logic describing computation intensive parts of vision algorithms, also embedded processors shall be utilized. Moreover external memories are included on that layer.
The communication between Vision Algorithms Wrapper Layer and Host PC Software Stack Layer is being established at the host operating system application layer. The vision algorithm wrappers are written in C/C++ and utilize pre-compiled libraries and/or API’s of the host OS in order to transfer data to the Channel Layer.

In a similar way the communication between FPGA Hardware Layer and FPGA Software Stack Layer is being established exclusively on the FPGA device and depends on the characteristics of hardware architecture. There are two possible scenarios for the architecture that affect the aforementioned communication. These scenarios are depicted in Figure 5. The choice for the implemented architecture for the SPARTAN system will be based on the results of HW-SW partitioning of visual algorithms and the selection of the Channel protocol.

For the choice of the protocol for the channel, several parameters of the system architecture shall be taken into consideration. We extinguish the system communication needs in terms of required bandwidth. Typical solutions for this purpose are UART, PCI, USB, Firewire, and Ethernet interfaces. Based on SPARTAN specifications, the most candidates among them is the Ethernet and PCI for high bandwidth connection links and RS232 for low-bandwidth links, respectively.

This choice, depicted in Figure 6, is based on the fact that those protocols are well supported by software libraries both in the host pc and the FPGA. In addition to that, these protocols are well documented by Xilinx tool-flow, which will be used in SPARTAN project.

However the JTAG protocol, except for bit-stream configuration on FPGA, shall be used in debugging time. In general the JTAG offers the following uses:
- Debugging: JTAG is used for accessing sub-blocks of ICs.
- Boundary Scan Testing: JTAG manipulates the scan chain external interface (inputs and outputs to other chips) in order to test for certain faults. By using JTAG to manipulate its internal interface (to on-chip registers), the combinational logic can be tested.
- Debugging at run-time (during system integration step): By programming and read-back the configuration bit-stream, it is possible to find potential errors/upsets occurred at run-time. Due to the inherent constraints for fault masking posed by the target application domain (space application), such a mechanism can provide sufficient fault detection and correction (through partial and/or dynamic reconfiguration). Note that debugging at run-time affects only the development step of the prototype in order to ensure that the entire system works properly.

V. CONCLUSION

This paper discusses the computer vision algorithms that will be implemented during the SPARTAN project. The employed implementation methodology was described and the early design analysis of SPARTAN system was presented.

ACKNOWLEDGES

The authors would like to thank Prof. A. Gasterator, Dr. L. Nalpantidis, Mr. I. Kostavelis and Mr. E. Boukas from DUTH, Greece for their remarks and expert opinion on the discussed CV algorithms.

REFERENCES