Smart multispectral image acquisition and multi-channel image processing with programmable System on Chip devices

Mathias Schellhorn[a], Richard Fütterer[a], Gunther Notni[a]
[a] Ilmenau University of Technology, Faculty of Mechanical Engineering, Department of Quality Assurance and Industrial Image Processing, P.O. Box 10 05 65, 98684 Ilmenau, Germany.

ABSTRACT

Through the acquisition and processing of several spectral channels within the multispectral data, the demands on signal processing and data handling increase enormously. With the help of intelligent signal pre-processing on programmable system on chip platforms (pSoC), captured data can be corrected and evaluated directly after image acquisition. PSoC combine the advantages of freely programmable logic (FPGA) and sequential processor systems (ARM technology) and significantly increase the integration density of embedded image processing systems. However, the design effort for these systems is increasing strongly, so that hardware/software co-design approaches must be used for implementation. The paper covers the design methodology, the implementation and the evaluation of multichannel acquisition systems using multispectral image sensors as an example.

Keywords: programmable system on chip, spectral imaging

1. INTRODUCTION

The research results presented here are based on the research topics "Hardware / Software Co-design methods for multidimensional image processing and image stack processing on system on chip platforms" of the InnoProfile project "QUALIMESS Next Generation" (03IPT709X). As research milestones, two smart spectral imagers with embedded pre-processing on the integrated pSoC were developed.

For the first demonstrator, the previously existing research setup of a filter wheel camera developed in the previous project QualiMess was revised and extended with a programmable SoC module. The aim was to further increase the precision of the spectral image while minimizing the latency between image acquisition and spectral image output. Additional hardware resources are used to enable embedded pre-processing of the spectral data. The use case is a Principal Component Analysis (PCA). The PCA provides uncorrelated linear combinations of the observed wavelengths and is an important pre-processing step for feature extraction and data reduction of multi- and hyperspectral images. These results were evaluated in a second demonstrator using the example of a spectral one-shot sensor with a different processing sequence and reduced spectral dimension.

This paper provides an overview of the entire project duration. For a more detailed look at the individually addressed results, reference is made to [1], [2], [3] and [4].

2. PRORGAMMABLE SYSTEM ON CHIP DEVICES FOR MULTIDIMENSIONAL IMAGE PROCESSING

Heterogeneous FPGA SoC or programmable SoC have been available from various hardware manufacturers since 2011. These devices combine one or more hardware-based processors (hard IP core) with a freely programmable FPGA structure closely linked on a die. The processor system (PS) and programmable logic (PL) are tightly linked by special control buses and dedicated high-speed ports for the fast exchange of large amounts of data. The spectrum of these components is extended by high-speed ports for the implementation of powerful interfaces as well as additional dedicated hardware components such as memory controllers, clock managers, DSP elements and floating point computing units.

The following is a rough summary of the advantages of using heterogeneous FPGA SoC:

- Reduced space requirements on the Printed Circuit Board (PCB) by concatenated hardware in a single chip

Photonics and Education in Measurement Science 2019, edited by Maik Rosenberger, Paul-Gerald Dittrich, Bernhard Zagar, Proc. of SPIE Vol. 11144, 1114404
© 2019 SPIE · CCC code: 0277-786X/19/$21 · doi: 10.1117/12.2530820

- Parallel hardware acceleration of embedded algorithms
- Energy efficiency compared to pc-based processing
- Broad support of different I/O standards
- Flexible use of the same hardware by using different hardware initializations or reconfiguration at runtime
- Extended hardware lifecycles due to long industrial availability of the components and the possibility of hardware updates

These advantages arise obviously from new challenges.

- Implementation requires profound hardware and software knowledge.
- Mastery of different implementation tools is necessary
- High integration density requires a precise timing and functional analysis
- Partitioning of functionalities implemented in hardware and software is essential for the performance of the overall system.

The last point in particular is a decisive factor for a successful implementation, partitioning takes effect early in the development process and influences the entire implementation.

2.1 Selection of a PSoC device for the implementation of embeded multidimensional image processing

The investigations within QualiMess next Generation included performance analysis and evaluation of the available programmable SoC. The aim was to create a systematic of different pSoC components and to identify essential comparison features with special consideration of the application of multidimensional image processing algorithms. For this purpose, market research on the available pSoC systems and a comprehensive hardware evaluation were carried out in 2015.

The tight memory link between PS and PL was defined as an important criterion because access to the shared data (image data) is often the bottleneck for optimal processing. Altera's pSoC (now Intel) and Xilinx offer the necessary connections combined with powerful PS computing power. Microsemis SoCs, typically in the lower to mid-range range, provide only rudimentary support that is not sufficient for more complex image processing.

As a result, the fully programmable SoC family Xilinx Zynq®-7000 with the Zynq-7020 as the target device was selected. The Zynq family combines an ARM® dual-core Cortex A9 core, programmable logic and essential peripheral functions. It is supported by a large number of IP modules and a tool infrastructure, which also makes it possible to realize very demanding tasks. Versatile prefabricated IP modules (Intellectual Property) and a tool infrastructure, which also enables the implementation of demanding tasks, support the development process. [5]

2.2 Development of a pSoC hardware platform as a versatile research setup

Based on the pSoC selection, a modular target platform was developed. The hardware should allow the connection of one or more sensors of different types (CCD, CMOS, variable bit depths, variable spectral sensitivities). For embedded image processing, an external memory had to be available that supports the required bandwidths for the various applications. In addition to the corresponding logic and computing power in PL and PS, other peripheral devices such as lighting or motion devices had to be connected. For this purpose, appropriate standard interfaces had to be available or be able to be connected via freely programmable IOs. After processing in the system, the images or data should be able to be transferred to display devices or PCs via appropriate image processing interfaces.

In order to accelerate the development process, industrial pSoC modules were used. These modules already deliver critical power supplies as well as impedance-controlled memory connections with correspondingly large memory modules. The IOs can be connected via commercially available connectors. The modules are therefore used together with a baseboard that provides additional peripherals and interfaces. Modules from Trenz GmbH and later from Enclustra were used.

Based on the catalogue of requirements, a block diagram was designed for the implementation of the baseboard. Based on this, a hardware board was created in several iteration steps, which forms the basis for the hardware demonstrators to be created in the project.

3. HARDWARE-SOFTWARE-CO-DESIGN OF MULTIDIMENSIONAL ALGORITHMS ON PROGRAMMABLE SYSTEM ON CHIP

The use of pSoC significantly increases the integration density of embedded image processing systems. The design effort for these systems is markedly increased, which makes a hardware-software co-design necessary. According to the system specification, the system is divided into hardware, software and interfaces. In order to avoid arbitrary partitioning, experience with the algorithms to be implemented is required. One focus of the investigations in the project was the classification of image processing algorithms for multidimensional and multi-channel image processing.

3.1 Systematics for separating algorithms that can be implemented by software or hardware

In the previous QualiMess project a general systematic for the separation of host-based and camera-based functions was developed. Since the system is based on pure FPGA structures in the camera system, it must be adapted to the current pSoCs. With pSoCs, the PL can be more closely linked to sequential processing in PS (Table 1), so that algorithms can be partly subdivided into PL and partly into PS and processed (co-processing). The bottleneck is the interface between PS and PL. Algorithms must therefore be critically analyzed for parallelizable components.

Table 1. Comparative overview between the processing architectures in PL and PS

	Programmable logic (PS)	**Processing system (PS)**
Processing	massive parallel	sequential
Processing speed	Extremely fast, different clock domains possible, real-time processing	fixed processor clock, many dependencies, interrupt controls or prioritizations lead to non-deterministic behavior
Limitations	Degree of parallelization, bit depths and local memory influence resource utilization	Essential limitation only by program memory, processing stack and heap
Programming	Hardware description languages, Register Transfer Level (RTL), graphical and (depending on manufacturer) high-level languages	Standard high languages like system C, C++, C#, ...

On this basis, we divide the image processing algorithms into four categories:

- **Data-independent algorithms** are characterized by the independent calculation of each data element (usually pixels). This includes point operators such as image transformations using the lookup table (LUT). But also the combination of several images to one output image via a linear linkage of the respective pixels in the input images. These algorithms are therefore ideally suited for calculation with hardware processors within the PL and can be easily parallelized. Limitations result from the memory bandwidth and the data transfer effort.
- **Data linking algorithms** include algorithms that require neighboring elements to be included in the calculation of data elements. Local buffering of one or more image lines is required. Nevertheless, they have a high parallelization potential, but must be optimized with regard to memory latency and multiple data usage.
- **Data-exchanging algorithms** are algorithms that require additional complex calculations (e.g. integral equations) that cannot be mapped in hardware. By an interaction of parallelizable hardware coprocessors and sequential calculation, however, noticeable calculation accelerations can be achieved.
- **Data-dependent algorithms** are characterized by extensive data dependencies and communication. Therefore, they are poorly suited for porting to the PL, since even enormous programming and optimization efforts can only achieve modest accelerations. An example of this would be error diffusion.

3.2 Assessment of the available development environments

For the design entry into the FPGA development process of the PL different approaches have been established, which were examined during the project duration. This includes the classical hardware description languages, high-level synthesis development environment (HLS) and graphical development environments.

Xilinx provides the Vivado development environment for the Zynq-7000 pSoc used in the project. This provides the environment in which the hardware platform is created. Xilinx uses an IP-Core based (Intellectual Property) approach. IP cores are self-contained function modules that can be integrated several times into the design. The cores are already fully specified and tested and can usually be further specified for use via configuration parameters. This approach saves development time and programming work, since large parts of a design are already available in part drafts that can be used in different systems (design re-use). In this way, projects can be granulated into individual function blocks that are reusable and easy to integrate. The IP catalogue provided by the manufacturer can also be extended with 3rd party cores, i.e. purchased IP cores, as well as self-written cores.

The design entry for most IP-Cores is usually done via hardware description languages (HDL) like VHDL or Verilog. HDL are used to describe the structure and behavior of digital circuits. In addition, syntax and semantics of HDL include notations to express time sequences and instantiations required in the hardware module.

Another option is high-level synthesis (HLS), for which Vivado includes the Eclipse-based Vivado HLS development environment. Vivado HLS converts algorithms written in high-level languages (C++, System C) to hardware specifications (e.g. Verilog and VHDL) of the Register Transfer Layer (RTL). The development environment analyzes the code to identify dependencies and parallelism and to map them to hardware.

However, since the input description is abstracted, the hardware optimization is passed from the designer to the HLS compiler, which can lead to inefficient implementations. The parallelization can usually be influenced by providing 'hints' (e.g. Pragma statements in C-based HLS) to the compiler. These hints usually also require corresponding design competence at the hardware level.

In the project, the IP-based design process of Xilinx was taken up and an interface concept for all IP-cores for uniform integration and interchangeability was defined. Vivado HLS is used for the creation of IP cores for data-independent, data-linking and partly also data-exchanging algorithms. For time-critical IP cores (e.g. special external interfaces) the IP development is carried out directly in VHDL.

3.3 Design of an framework for the implementation of multidimensional algorithms

As a basis for the following implementations, the design and implementation of a basic infrastructure had to be realized. Infrastructure in this context represents a basic framework for pSoC systems, a architecture for multidimensional image processing applications. This required the implementation of image acquisition and control of image sensors, intermediate storage of image stacks, variable image processing pipelines and output via standard interfaces.

For the development of the architecture different approaches were examined. Typical image processing pipelines in FPGA devices work according to the well-known EVA principle (input processing output). The data is sequentially routed through the individual processing units (or IP cores) and can thus be processed in parallel.

As explained in the introduction, pSoC exchang data between PL and PS via external RAM. In the case of the Xilinx pSoC used, this is done by using prefabricated DMA (Direct Memory Access) or especially in the image processing area via VDMA (Video Direct Memory Access) IP cores. These allow the storage or reading of sequential image data streams into external RAM memories with a management of up to 32 image memory addresses.

In the FPGA part of the pSoC various manufacturers use the AXI bus (Advanced Exensible Interface) as standard for the connection of hardware components. The AXI4 was specially developed for the exchange between digital subsystems and ARM processors and is a widely used on-chip interface standard. The standard defines:

- **AXI4 Full** for the communication of components via address information. The high-speed interface is specially designed for memory access with burst widths of 256 data sets at 32 or 64 bits.
- **AXI4 Lite** as a communication and control channel that requires individual addressing for each data packet.
- **AXI4 Stream** as a point-to-point connection between components via a 4-phase handshake without address information. [6], [7]

The VDMA IP-Core is equipped with AXI4 stream ports which can be integrated into the image processing chain. The connection to the external memory is established via AXI4 full and the hardware-cast Dynamic Memory Controller. Control is via AXI4 Lite and must be triggered via software routines in the ARM.

With the help of these IP cores and external memory, different processing chains can be realized. A simple image buffer with variable image output can be realized by integrating the VDMA core. With the same resources in different configurations, the processing chain can be converted into a ring structure that allows the same chain to be run through several times starting from the image memory. This allows variable pipeline processing, but at the expense of performance and overall latency. The use of display interfaces such as HDMI, which require fixed frame rates, is also critical in this context.

Therefore a memory decoupling of the actual image processing from the image input output was chosen as approach for the framework. For this purpose, the image processing pipeline is separated and connected to the external memory via VDMAs. The image input / output path contains only basic image restoration and error correction algorithms after image acquisition. The images are then transferred to fixed memory areas in the RAM using the pixel clock of the sensor / camera. At the same time, images are read from an output area and transferred to the host via the interface. At the same time, the system has a scalable number of processing pipelines that buffer their results in RAM. By using AXI4 Stream Interconnect IPs in switch operation, the processing chain and the processing of individual IP cores can also be controlled. The rough overview of the architecture is shown in Figure 1 as a block diagram.

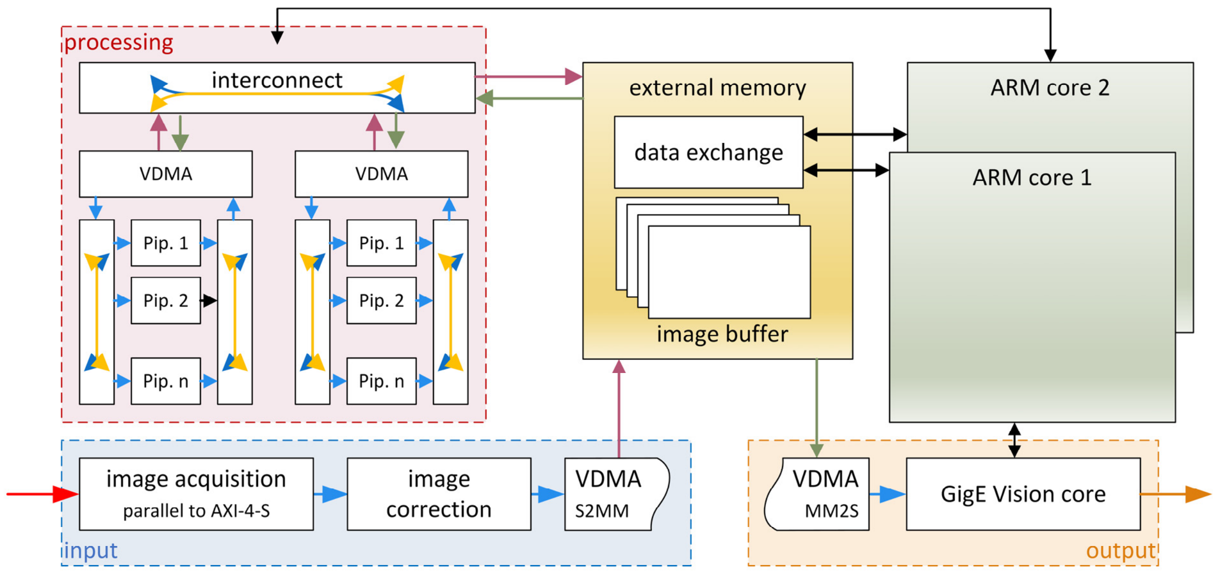

Figure 1. Simplified schematic representation of the framework underlying the implementation strategy. [3]

The framework was implemented as a block design with the tools of the Vivado Design Suite. Additional hierarchies were set up for the individual parts of the block design to additionally structure the project.

4. IMPLEMENTATION OF MULTIDIMENSIONAL IMAGE PROCESSING ALGORITHMS

Based on the framework, a Principal Component Analysis (PCA) algorithm was implemented on the programmable SoC platform developed in the project. The sequential algorithm was divided into algorithm segments, implemented and integrated into the framework.

Probably the most commonly used multivariate statistical technique used by almost all scientific disciplines. It reduces the dimension of the data by eliminating noise and redundancy. This aligns the data to the main components (PC) where the variance of the data is maximum. The PCA represents an Eigendecomposition (singular value decomposition) of the input matrix or its correlation or covariance matrix. The resulting Eigenvectors are sorted in descending order according to their magnitude and entered into the factor matrix. Consequently, the first columns of the factor matrix contain the Eigenvectors with the highest Eigenvalues, which also describe the largest part of the total variance of the input data. During the subsequent data reduction, relevant information and noise can be separated by meaningful reduction of the main components and subsequent projection of the input data. This allows hidden information to be identified and patterns that are not clearly visible in the original data to be highlighted. A good overview of the mathematical backgrounds is given in [8]. Figure 2 shows an overview of the individual calculation steps required.

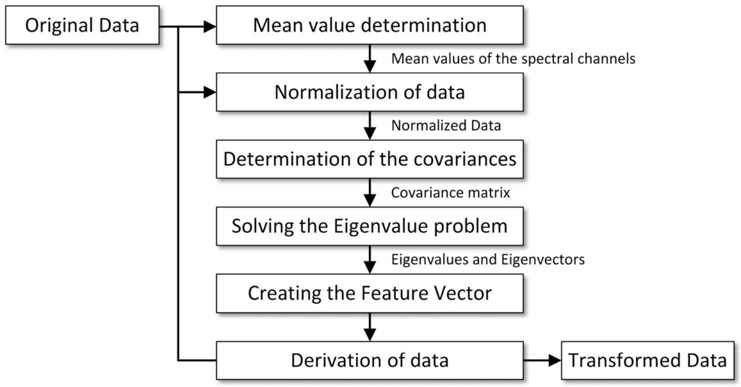
Original Data
Mean value determination
Mean values of the spectral channels
Normalization of data
Normalized Data
Determination of the covariances
Covariance matrix
Solving the Eigenvalue problem
Eigenvalues and Eigenvectors
Creating the Feature Vector
Derivation of data
Transformed Data

Figure 2. Successive calculation steps of a PCA [4].

When applied to the spectral data of a multispectral camera, the individual images, each corresponding to an image acquisition with a specific wavelength, must be calculated against each other. This assumes that all images of the spectral data set are stored temporarily for the entire computation time. This requires the image data to be stored in the external memory. In addition, the intermediate results and the derived image data must be generated, which are also stored in the memory. The memory connection as well as the memory accesses are therefore essential for the efficient and fast calculation within the PL. Some development decisions relevant for implementation are outlined below. For in-depth explanations, please refer to [2], [3] and [4].

4.1 Calculation of the mean values of the spectral image data (data-independent algorithm)

The functionality was implemented in VHDL as IP-core with an AXI4 Lite interface. The 8 bit grey values of the pixels transmitted one after the other in the data stream of the AXI4 stream are summed in a hardware accumulator (DSP slice). The last pixel triggers an interrupt so that the PS can access this value via AXI4 Lite. In addition, the image dimensions are determined by the synchronization signals themselves and made available via AXI4 Lite registers. Thus, the following calculation section can be dynamically adapted to different acquisition systems.

In contrast to the other calculation steps, the average value of the individual images is not calculated in the processing branch, but directly during data input. The IP core was added directly after image acquisition. The actual mean value is calculated from the transferred pixel sum within the PS. Both divisions and multiplications require valuable hardware resources. Since the determination of the mean value is not time-critical compared to the rest of the calculation, the division is calculated in the PS in a resource-saving way.

The mean values are stored as IEEE 754 coded floating point numbers in a parameter block on the external memory. Here, a memory area was organized in a uniform format that is available as shared memory for the PS and all relevant IP cores of the PL. This means that pSoC-wide work can be carried out on a unified database. However, access rights and times must be observed.

4.2 Calculation of covariances and creation of the covariance matrix (data-linking algorithm)

The calculation of the individual covariances is the most time-consuming part of the algorithm, since the original data must be computed with each other. For this purpose, a hardware IP-core was developed and programmed in Vivado HLS.

The first iteration of the IP-core calculates exactly one covariance between two images. The IP-core reads the image data autonomously from two RAM addresses via an AXI4 full interface. These are calculated pixel by pixel. The data is normalized live during the calculation. The RAM addresses and the mean values are transmitted by the PS via an AXI4-

Lite interface. At the end of the calculation, the respective covariance is also read out via AXI4-Lite. The IP core itself is resource-efficient, but requires a high configuration effort and a high memory bandwidth for several calculation runs. The sequential call of the IP core led to only moderate calculation times of 80 ms per core run, which corresponds to a total calculation of 6,240 ms.

In order to achieve reduced computing times, the parallelization in the core was increased and at the same time the multiple readout of single images was minimized. The core was implemented in such a way that as many covariances as spectral channels exist, are calculated simultaneously. Due to the symmetry of the covariance matrix (mirrored into the main diagonal), not all calculations are necessary depending on the filter level. To take this fact into account, a corresponding configuration parameter was implemented.

The RAM addresses of the image data are already transferred during the initial configuration, so that only the respective mean value and the current filter level have to be transferred to carry out the calculation. The calculations are carried out in parallel. Since a fixed number of calculations must be implemented in hardware, a maximum of 12 parallel computing units were realized. This corresponds to the number of spectral channels of the filter wheel camera used as the test setup. The calculations run in parallel and are called in loops according to the number of pixels.

Through optimization, the computing time could be reduced to an average of 58ms per single core call. With 12 core runs required, this corresponds to a total calculation time of 697ms.

4.3 Eigendecomposition of the covariance matrix (data-dependent algorithm)

The Eigendecomposition is the most complex calculation step within the overall algorithm, but can be calculated relatively quickly and requires only the covariance matrix itself. The problem was therefore implemented within the PS. The eigenvalue problem can only be solved after the last covariance has been calculated. For this purpose, an iteration method based on QR decomposition with Householder transformation was implemented in the PS. The calculated eigenvalues and eigenvectors are stored as matrices in the heap memory of the processor. The vectors are sorted by size as feature vectors.

4.4 Projection of the original data, derivation of the aligned main components (data linking algorithm)

The data is derived via a hardware IP-core programmed in Vivado HLS. The original data is read from the external RAM via a full AXI4 interface. For the configuration, the necessary RAM addresses of the original data, the corresponding mean values, the feature vector and the target RAM addresses for the calculated PC are also read from the shared memory via AXI4 full.

In the IP, additional to the image data, the parameter block must be organized internally. In addition to unsigned integer values (RAM addresses), float values (mean values and eigenvectors) are also required, which are stored within the RAM as 32-bit values. This means that, in addition to memory access, a reconversion according to IEEE 754 is also required. For this configuration, an additional AXI4 full interface was implemented, which enables burst access to the RAM. The internal memory was implemented using block RAM, which was segmented according to parameter ranges using compiler pragmas. The individual block RAM areas operate as dual-port RAMs. This enables parallel access to the individual registers, which minimizes computing time.

A parallel calculation of several PCs requires an enormous amount of resources, especially look-up tables. For the implementation in the filter wheel camera, where the Core would have to calculate 12 PCs, the utilization of the Core alone would amount to over 80 percent of the pSoC's resources. Since the first main components already contain most of the information of the spectral range, only a parallel calculation of three PCs was carried out. Parallel to the calculation, the mean value of the pixel values is calculated for each resulting image. The average values can be used as an abort criterion for several runs of the core.

5. CONCLUSION AND FUTURE WORK

In this summary paper, the design of pSoC platforms was explained using the example of the implementation of a PCA algorithm. The general design process for the embedded implementation of algorithms on pSoC was described. Important design hints for architectures are given, especially with regard to applications in spectral image processing with multidimensional data sets. The memory-decoupled realization of image acquisition, image stack processing and image output is necessary to be able to provide the data of the individual spectral channels independently of the sensor principle. The processing branches can therefore process image data block by block independently of the sensor clock.

In addition, a model was proposed that uses a common parameter block for configuration data in the form of a shared memory. This ensures a common database at runtime. In addition, the configuration effort of the PS is reduced since IP-cores in the PL load their configuration data themselves and can therefore work independently. For hardware-software partitioning, a general classification for algorithms with regard to their access to the database was proposed.

For the research, an implementation was realized that works independently of the spectral recording principle. The results were evaluated in the form of a filter wheel camera with 12 spectral channels and a one-shot sensor with 9 spectral channels. Independent of the data dimension of the input data, a dimension reduction to three principle components was realized. The system also offers to calculate low-order PCs, which, however, requires additional calculation time.

As the number of dimensions within the data increases, a limit value for parallelization must be selected in order to make sensible use of the limited resources. This decision must be made depending on the algorithm. For the evaluation of the spectral series a reduction of the data dimension to 3 components is often sufficient, since a large part of the information is already contained here. The restriction to three components also has the advantage that these can be represented as colors. A simple possibility would be to interpret the first three components as RGB channels of an image.

Further work includes the implementation of a partial reconfiguration of the processing in the PL. In this way, parts of the hardware can be reconfigured at runtime and resources can be used more efficiently for processing the individual algorithm components and a higher degree of parallelization.

ACKNOWLEDGMENT

We thank the federal ministry of education and research for the strong support of this work. The work is related to the project “Qualimess next generation” (03IPT709X).

REFERENCES

[1] M. Rosenberger and R. Celestre, “Smart multispectral imager for industrial applications,” in *IST: 2016 IEEE International Conference on Imaging Systems & Techniques : October 4-6, 2016, Chania, Crete Island, Greece : proceedings*, Chania, Greece, 2016, pp. 7–12.

[2] M. Schellhorn *et al.,* “Smart parallel spectral imager based on heterogeneous FPGA system on chip,” (en), *Engineering for a Changing World: Proceedings; 59th IWK, Ilmenau Scientific Colloquium, Technische Universität Ilmenau, September 11-15, 2017*, vol. 59, 2017, no. WS.3.01, https://www.db-thueringen.de/receive/dbt_mods_00033247, 2017.

[3] M. Schellhorn, R. Fütterer, G. Notni, and M. Rosenberger, “Programmable system on chip implementation of a principal component analysis for preprocessing of multispectral image data acquired with filter wheel cameras,” in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXIV: 17-19 April 2018, Orlando, Florida, United States*, Orlando, United States, 2018, p. 63.

[4] M. Schellhorn and G. Notni, “Optimization of a Principal Component Analysis Implementation on Field-Programmable Gate Arrays (FPGA) for Analysis of Spectral Images,” in *2018 International Conference on Digital Image Computing: Techniques and Applications (DICTA): Canberra, Australia, 10 December-13 December 2018*, Canberra, Australia, 2018.

[5] Xilinx Inc., *UG585 Zynq-7000 All Programmable SoC - Technical Reference Manual (v1.12.2).* [Online] Available: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf. Accessed on: Jun. 07 2019.

[6] J. Reichardt, *Lehrbuch Digitaltechnik: Eine Einführung mit VHDL*, 2009.

[7] Xilinx Inc., *UG761 AXI Reference Guide, v13.1.* [Online] Available: https://www.xilinx.com/support/documentation/ip_documentation/ug 761_axi_reference_guide.pdf. Accessed on: Jun. 07 2019.

[8] L. I. Smith, “A Tutorial on Principal Components Analysis,” *Cornell University, USA*, vol. 51, p. 52, 2002.