A new real-time system for image compression on-board satellites

Kristian Manthey(1), David Krutz(1), Ben Juurlink(2)

(1)German Aerospace Center (DLR)
Institute of Optical Sensor Systems, Optical Sensors and Electronics
Rutherfordstraße 2
12489 Berlin-Adlershof, Germany
Email: {kristian.manthey|david.krutz}@dlr.de

(2)Technische Universität Berlin
Embedded Systems Architecture
Einsteinufer 17
10587 Berlin, Germany
Email: b.juurlink@tu-berlin.de

INTRODUCTION

Remote sensing sensors are used in various applications from Earth sciences, archeology, intelligence, change detection or for planetary research and astronomy. Disaster management after floodings or earthquakes, detection of environmental pollutions or fire detection are examples of countless number of applications. The spatial as well as the spectral resolution of satellite image data increases steadily with new technologies and user requirements resulting in higher precision and new application scenarios. In the future, it will be possible to derive real-time application-specific information from the image on-board the satellite also based on high-resolution images. On the technical side, there is a tremendous increase in data rate that has to be handled by such systems. While the memory capacity requirements can still be fulfilled, the transmission capability becomes increasingly problematic.

In this paper, an image compression architecture with region-of-interest support and with flexible access to the compressed data based on the CCSDS 122.0-B-1 image data compression standard is presented. Modifications to the standard permit a change of compression parameters and the re-organization of the bit-stream after compression. An additional index of the compressed data is created, which makes it possible to locate individual parts of the bit-stream. On request, stored images can be re-assembled according the application's needs and as requested by the ground station. Interactive transmission of the compressed data is possible so that overview images can be transmitted first followed by detailed information for the regions of interest (ROIs).

IMAGE DATA COMPRESSION ON-BOARD SPACECRAFT

The first known satellite with on-board image compression, was SPOT-1 (1980) [6]. A transform-based compression algorithm was first used for the PHOBOS (1988) mars exploration missions. The algorithm used a discrete cosine transform (DCT) for spatial decorrelation, followed by scalar quantization and fixed length coding. Compression was performed off-line on a Z80 [6, 13]. In the following years, the data rate of high-resolution systems has increased rapidly. The JPEG standard was approved in 1992 and used in many remote sensing missions with moderate data rates. At that time, high-resolution systems such as IKONOS (1999), QuickBird (2001) or WorldView-1 (2007) are using relatively simple algorithms such as differential pulse code modulation (DPCM) in order to perform image compression in real-time [1].

In 1991, even before the JPEG standard was approved, CNES developed a JPEG-like compression ASIC; which is capable of 4 Mpx/s real-time compression. For SPOT-5 (2002), a compression architecture targeted for Earth observation was developed [10]. SPOT-5 contains three instruments producing up to seven data streams; each up to 128 Mbit/s. A proprietary adaptive DPCM image compression algorithm was used in IKONOS (1999), QuickBird (2001) and WorldView-1 (2007). The compression application-specific integrated circuit (ASIC) achieved the operating rate of 22 Mpx/s [1]. EADS Astrium developed Compression Recording and Ciphering (CoReCi) which is used for SPOT-6 (2013), SPOT-7 (2014) and KazEOSat-2 (2014).It uses multiple dedicated compression ASICs called Wavelet Image COmpression Module (WICOM) with a speed of up to 25 Mpx/s.CCSDS Wavelet Image COmpression Module (CWICOM) is an image

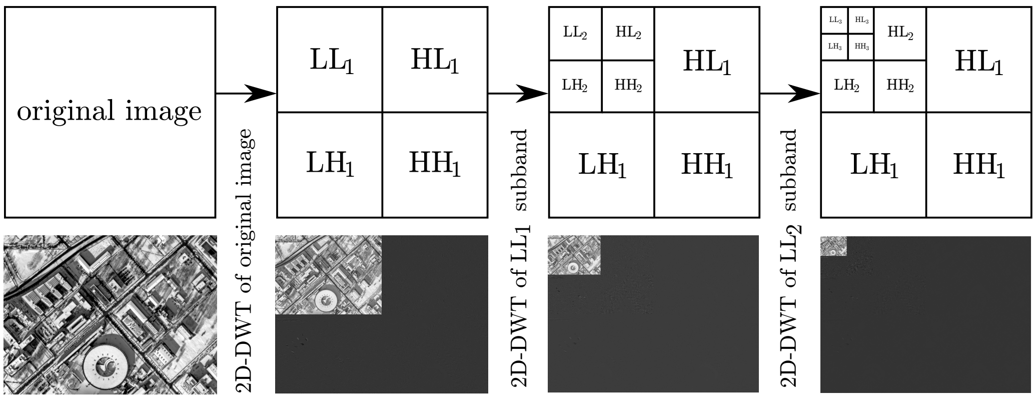

Fig. 1: Three-stage two-dimensional DWT.

compression ASIC which implements the CCSDS 122.0-B-1 standard and supports both lossy as well as lossless image compression at a data rate of up to 60 Mpx/s.

In recent years, FPGAs have been increasingly used for space applications. The space environment delivers challenging requirements regarding reliability, power consumption, operating temperature range, vacuum, radiation, shock resistance and component qualification. GRACE (2002) and FedSAT (2002) were the first missions using an early Xilinx space grade FPGA [4]. The FedSAT system is able to compress pixels at a rate of 43.8 Mpx/s. An overview of FPGA-based image compression systems is presented in [13]. The author also proposes an architecture for on-board image compression. The design and implementation of the FORMOSAT-5 (2015) Remote Sensing Instrument (RSI) is described in [8] and [9]. FORMOSAT-5 is an optical satellite with a ground sample distance (GSD) of 2 m. The total output data rate of the instrument is 970 Mbit/s (PAN + 4×MS). Three space grade Xilinx XQR5VFX130 are used for online image compression [11]. One FPGA is used for PAN processing and two FPGAs are used for MS processing. The compression system uses 24 external SRAM chips, each 1 Mbyte. An implementation of the CCSDS 122.0-B-1 standard used for Proba-V (2013) and EnMAP (2015) is presented in [7]. The estimated data throughput is 173 Mbit/s at 66 MHz.

CCSDS 122.0-B-1 IMAGE DATA COMPRESSION

CCSDS 122.0-B-1 Image Data Compression standard [2, 12] is a single-band compression technique and has been recently used for image data compression on-board spacecraft. It can compress 16 bit signed and unsigned images in lossless as well as in lossy mode. A CCSDS 122.0-B-1 encoder consists of two parts: The discrete wavelet transform (DWT) module and the bit-plane encoder (BPE).

At first, the DWT module applies a three-level two-dimensional DWT on the input image as shown in Fig. 1. Two specific wavelet filters are provided by the standard: The CDF(9/7) wavelet transform referred to as “Float DWT” and a reversible integer approximation of that transform referred to as “Integer DWT”. The Float DWT is intended to be used for lossy compression, whereas lossless compression can only be achieved with the reversible Integer DWT. In the case of the Float DWT, limited precision of the coefficients’ floating point representation and a conversion to the nearest integer after transform lead to some loss of information.

The DWT module forms a hierarchy of wavelet coefficients. A *block* is a group of one DC coefficient and the 63 corresponding AC coefficients (3 parents, 12 children, 48 grandchildren). A block loosely represents a region in the input image. For the BPE, the blocks are further arranged into groups: A segment is a group of S consecutive blocks, where $16 \leq S \leq 2^{20}$. Segments are encoded independently and are further partitioned into *gaggles*, which is a group of $G = 16$ consecutive blocks.

Once all coefficient are grouped, the BPE starts to encode the image segment-wise. The first step is to *weight* the coefficients if the Integer DWT is used. This is necessary to optimize the rate distortion [3]. Since the sub-band weights have been obtained empirically, the standard supports user-defined weights. Every sub-band has its own weighting factor w and the coefficients of each sub-band are multiplied by 2^w. Each segment starts with a *segment header* containing information about the current segment. The DC coefficients are coded in two's complement representation. The AC coefficients are coded in sign-magnitude representation. After the segment header is written, the DC coefficient are quantized with a quantization factor q that depends on the wavelet transform type and on the dynamic range of the wavelet coefficients. In a next step, DPCM is applied on the quantized DC coefficients and is followed by Rice Coding. After all quantized DC coefficients are encoded, some additional DC bit-planes may be refined. The next step is to encode the bit-depth of the AC coefficients in each block with the same DPCM method already used for the quantized DC coefficients.

The BPE encodes the wavelet coefficients, as the name suggests, bit-plane-wise and in decreasing order. For each bit-plane, the encoding process is divided into stages 0–4. In stage 0, remaining bits of the DC coefficients are coded (DC refinement). Stage 1–3 encode the AC coefficients' sign and the position of the *significant bit*, which is the highest non-zero bit. Stage 1 refers to the refinement of the parents coefficients. The same procedure is applied to the children coefficients at stage 2 and to the grandchildren coefficients at stage 3. Stages 1–3 produce words which are first mapped to symbols. The symbols are then encoded with variable-length code (VLC). All bits of a stage are written to the output bit-stream before the next stage is commenced, even though the optimal code for the VLC is determined by stages 1, 2 and 3. Once a AC coefficient is selected, Stage 4 encodes the AC coefficients' refinement.

EXTENSIONS TO THE CCSDS 122.0-B-1 STANDARD

CCSDS 122.0-B-1 neither supports ROI coding, multispectral compression, spectral decorrelation nor does it produce a bit-stream that can be re-assembled in any way. Compression parameters cannot be changed without re-encoding. ROI coding would be useful in scenarios where on-board classification, registration or object or change detection algorithms are used. If a certain event is detected or a matching object is found, the compression system might encode the corresponding area with a higher detail or lossless. If there are multiple ground stations with divergent downlink capability, a re-assembling of already compressed image data might be desirable in order to adjust the amount of data to the bandwidth of the transmission channel. Another application scenario for this approach are ground stations with different access rights to the resolution level or spatial areas of the images. Because the encoder generates an embedded bit-stream, the significance of each bit or the position of a block, segment or image region inside the bit-stream can only be determined by decoding. However, the algorithm is well suited for real-time image compression on-board spacecraft.

The basic idea of ROI encoding is to encode certain regions with a low distortion (lossless) and other regions with a higher distortion (lossy). A ROI mask contains the information about whether a certain region is of interest (or not). It is convenient to adjust the granularity of the ROI mask to a unit of information used in the compression algorithm. In this work, ROI coding is achieved by controlling the compression parameters segment-wise.

Scalability here means that the compressed image or parts of it can be re-assembled to achieve a particular image quality, spatial or spectral resolution in any area of an image without the need for re-encoding. Furthermore, the method will also be useful to efficiently build transfer frames for multispectral (MS) encoded data. A similar approach was presented in [5]. Scalability is achieved by a) using or modifying the compression algorithm in a way that different spatial or spectral regions of an image can be independently decoded, and b) creating an index of the encoded bit-stream so that the position of an image block is known. It is desirable a) to compress images with a region-specific spatial or spectral resolution based on a mask which is available during compression, and b) to assemble a transfer frame which probably contains an even coarser spatial or spectral resolution, i. e. to change compression parameters after compressed without re-encoding. Fig. 2 shows this concept on three images.

In order to manipulate the bit-stream after compression, this index must be stored in memory. On the one hand, this effectively reduces the compression performance. On the other hand, the index does not have to be transmitted to the ground station, since the re-assembling of the bit-stream and transfer frame generation is performed on-board the spacecraft. A segment size of $S = 128\,\mathrm{blocks}$ leads to an average index size of $1.19\,\%$ of the input image size for the examined images.

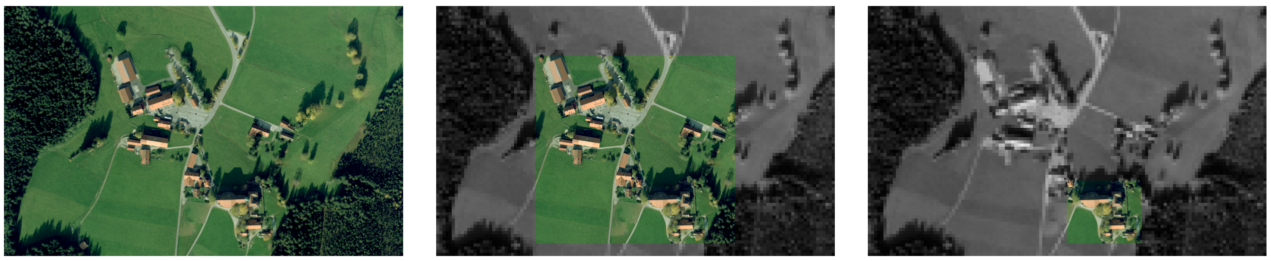
(a) Original image (b) Stored image (c) Downloaded image

Fig. 2: The same image at three different stages: a) before compression, b) after compression and c) as it is requested by the ground station.

HARDWARE ARCHITECTURE

In order to fulfill the high demands on the data throughput, which is approximately 200 Mpx/s for a single instance, and still achieve an effective and flexible compression, a pipeline approach is chosen. Fig. 3 shows the underlying structure of the proposed architecture.

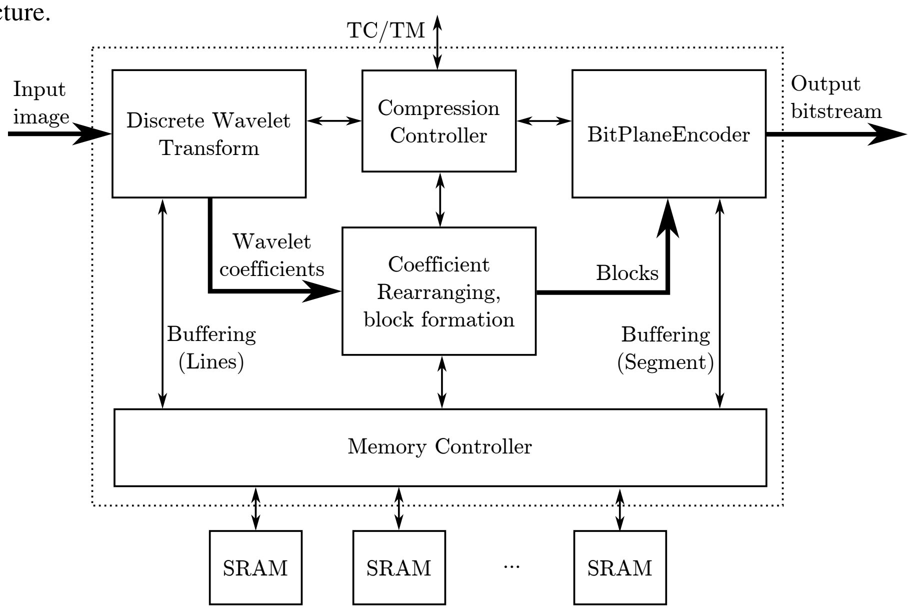

Fig. 3: Overview of the compression architecture.

The design of the compression system consists of five main modules. The Compression Controller (CC) communicates with the spacecraft via TC / TM and controls all modules of the system except the memory controller. Compression parameters and a ROI mask is supplied by the CC. The data path starts at the DWT module. The Coefficient Rearranging and Block Formation (CABF) module is used to re-arrange the coefficients from the ten sub-bands and to form the blocks necessary for the BPE.

The DWT module uses a line-based architecture for the three individual two-dimensional modules. Internally, each two-dimensional DWT module uses one row and two column transform modules. The memory demands of the two-dimensional DWT is predominated by the column transform, especially by the image width and the chosen wavelet kernel. For the “Float DWT” or “Integer DWT”, five or six lines must be cached respectively. If input values for two lines are to be processed, the corresponding entries stored in memory are read. At the end, the memory is updated for the next iteration. In case of the floating-point DWT, the corresponding lifting scheme is implemented directly. The memory requirements are as follows: Since up to six lines must be cached, up to six temporary values must be read and written every clock cycle. For an image width of 16384 px and an internal precision of 24 bpp, the total buffer size is 172032 px (504 kbyte). Assuming a clock frequency of 100 MHz, the total data rate of the memory – that is independent of the image size – is 3600 Mbyte/s.

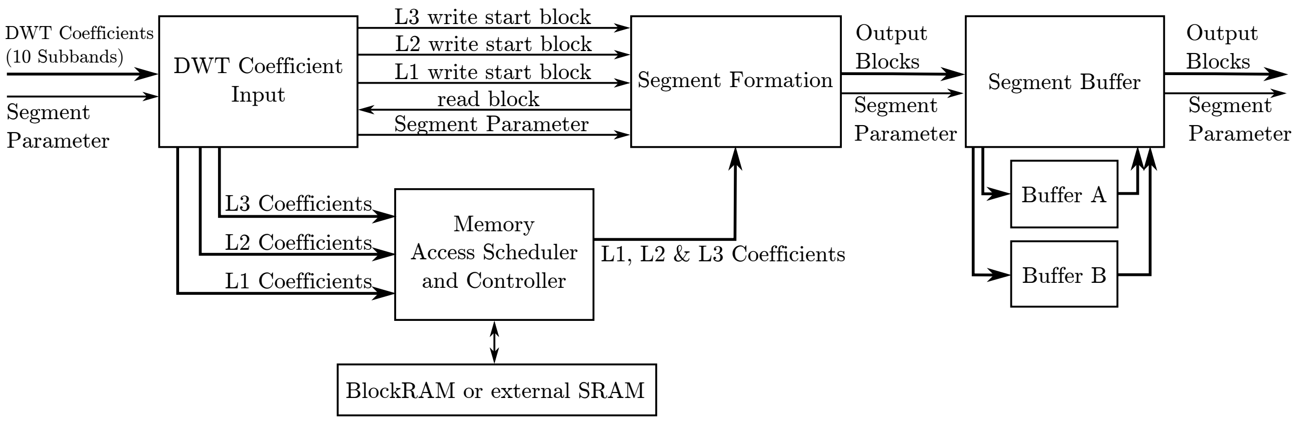

Fig. 4: Structure of the block rearranging module.

At the low-pass filter of the Float DWT, each four input values must be available around the current input value (prior and after). With a line-based architecture, this will cause an output latency of four rows for each decomposition stage. Every DWT module L_k decomposes a input signal LL_k into the four sub-bands LL_{k+1}, HL_{k+1}, LH_{k+1} and HH_{k+1}. The first output will arise a few clock cycles after the fifth input line has started. From now on, the DWT module produces output on every second line (due to the sub-sampling). When the first output arises in DWT module L_3, DWT module L_1 has already emitted 12 lines of wavelet coefficients. At this time, DWT module L_2 has emitted 4 lines of wavelet coefficients. Thus, at least $3 \cdot S_x$ blocks must be buffered for synchronization with the L_3 sub-band, where S_x denotes the maximum number of blocks in horizontal direction. A value of $4 \cdot S_x$ is chosen for this design in order to buffer more blocks. Thus, assuming a dynamic range of the wavelet coefficients of $d = 24\,\text{bit}$, the total amount of memory necessary for coefficient rearranging is as follows:

$$4 \cdot S_x \cdot 64\,{}^{\text{coefficients}}\!/\!_{\text{block}} \cdot 24\,\text{bit} = S_x \cdot 0.75\,\text{kbyte} \tag{1}$$

If the image has up to $S_x = 2048$ blocks in horizontal direction, the coefficient rearranging module requires 1.5 Mbyte of memory. Since all the coefficients need to be written and read only once, the data rate of the memory is twice the input data rate of the module. The input data rate of the module is $200\,{}^{\text{Mpx}}\!/\!_{\text{s}}$. Thus, the memory data rate (read and write) is $400\,{}^{\text{Mpx}}\!/\!_{\text{s}}$. Assuming a dynamic range of the wavelet coefficients of $d = 24\,\text{bit}$, the total data rate of the memory is $1200\,{}^{\text{Mbyte}}\!/\!_{\text{s}}$.

The idea of the entropy encoder is that the wavelet coefficients are read block by block, until the required number of blocks has been read and the entropy encoding process can start. Unfortunately, this is not the order in which the DWT module outputs the wavelet coefficients.A line-based DWT architecture has two problems: On the one hand, a CCSDS 122.0-B-1 encoder usually operates in stripe-based mode, i. e. the segment size is $S = S_x = N \times \lceil I_w/8 \rceil$ blocks, where I_w denotes the width of the image and N is an integer ≥ 1. In order to support regions-of-interest, the segment size must be variable or at least less than this value. In stripe-based mode, all blocks in any image line belong to the same segment. Otherwise, the DWT module generates coefficients belonging to multiple segments. On the other hand, the structure of the DWT module causes temporal delays of the wavelet coefficients in the higher decomposition levels L_2 and L_3 (see also Fig. 1).

The main task of the CABF module is to rearrange the wavelet coefficients for a block-wise output. It buffers the wavelet coefficients of the L_1 and L_2 decomposition levels and synchronizes its output to L_3. Thus, it also buffers coefficients belonging to another segment. Fig. 4 shows the structure of the block rearranging module. This input module receives the DWT coefficients of the three two-dimensional DWT modules and writes them to temporary memory. Therefore, it has to determine the block number a coefficient belongs to. The input and the output module share a circular list of memory block addresses: It signals the output module the position of the first LL_1, LL_2 and LL_3 block that has not been completely written. In turn, the output module signals the position of the last block that has been completely read. The size of the memory buffer depends on the number of blocks that must be buffered. This in turn depends on the length of the wavelet filter and the width of the image.

The segment buffer module provides blocks to the BPE module. A double buffering mechanism is used, so that the next segment can be written to the buffer, while the BitPlaneEncoder processes the last segment. The module buffers the weighted coefficients in sign and magnitude representation, each consisting of $d + 4\,\mathrm{bit}$, as well as the bit-depth of the AC coefficients of each block (5 bit). Assuming a segment size of $S = 128\,\mathrm{Blocks}$, the memory size will be $\approx 56\,\mathrm{kbyte}$.

The BPE is the entropy encoder of the compression algorithm and produces the individual bit-stream parts. It gets the input from the coefficient rearranging module. Without considering parallel execution mechanisms, it must be able to compress each segment in $32 \cdot S$ clock cycles. In order to achieve real-time compression, the individual modules must operate in parallel: A BPE compression control module reads the segment blocks from the coefficient rearranging module (segment buffer). Parameters necessary to write the segment header are also provided, so that the encoding process can start immediately. Whenever the segment buffer has collected an entire segment, it sends a request to the BPE compression control module. When the compression of the segment is finished, the module confirms this to the segment buffer. During operation, the BPE control module sequentially generates input data for the encoding modules.

Segment header, quantized DC coefficients, additional DC bit-planes and AC coefficient bit-depths are processed sequentially, while the stages 0–4 are processed in parallel. The reason for this is that stage encoding produces the major part of the compressed bit-stream and consumes the majority of the execution time. All encoding modules use a data valid mask (one bit for each data bit) in order to mark valid output bits, and two flags in order to mark the end of a segment or a bit-plane (end-of-segment and end-of-bit-plane).

The DC coefficient of each block is read from the segment buffer, quantized and transmitted to the QuantizedDC and BitDepthAC module.The module is also used to encode the bit-depth of the AC coefficients of each block. In both cases, one block is read from the segment buffer every clock cycle. All input values are first mapped to symbols. Then, the optimal code option for subsequent Rice coding is determined. Note that the architecture does not support the heuristic method presented in the CCSDS 122.0-B-1 standard, since it does not lead to a significant simplification of this architecture. During the determination of the rice parameter, the mapped symbols are cached in a small fifo. The depth of the fifo depends on the maximum gaggle size G and is set to $2 \cdot G$. Depending on the code option p, data is written uncoded in one stage or coded in two stages. In the next step, the additional DC bit-planes are read from memory and transmitted to the Additional DC Bitplanes module. DC coefficient refinement is performed depending on the DC quantization factor and BitDepthAC. In this step, some bits of the DC coefficients of all blocks are encoded. The order of the encoded bit was changed: Instead of sending the $(q-1)^{\mathrm{th}}$ most-significant bit of each DC coefficient followed by the $(q-2)^{\mathrm{th}}$ most-significant bit of each DC coefficient (and so on, until the $\mathrm{BitDepthAC}^{\mathrm{th}}$ bit of each DC coefficient), bits $q-1, \ldots, \mathrm{BitDepthAC}$ of the first DC coefficient is sent, followed by the corresponding bits of the second DC coefficient and so on. This reduces the number of memory accesses to the segment buffer.

After completion of the initial encoding of the segment, the encoder starts the stage encoding procedure. In this step, every block is read from the segment buffer for every bit-plane to be encoded. The encoding process runs for the bit-planes $b = \mathrm{BitDepthAC} - 1,\ \mathrm{BitDepthAC} - 2,\ \ldots,\ 0$. Now, stage 0–4 output is generated simultaneously. Stage 0 data – if valid – consists of a single bit (the corresponding bit of the DC coefficient). Stage 1, 2 and 3 data is generated from the current block status, the encoding parameters, the index of the current bit-plane, the number of the current gaggle and the signs of each AC coefficient. Stage 4 data consists of up to 63 bit, one bit for each AC coefficient that was selected in a previous bit-plane.

Depending on the image and the compression parameters, a segment consists of different bit-stream components or parts. The output module merges the bit-streams from up to eight encoding modules into a single output bit-stream. Furthermore, it creates the bit-stream index that is necessary to achieve scalability. The output module gets the information on the segment parts from the compression control module.

RESULTS AND DISCUSSION

The compression algorithm presented in the previous sections has been successfully implemented on reconfigurable hardware which is qualified for space applications.

Table 1: Data compression throughput for lossless compression and a segment size of $S = 128\,\mathrm{blocks}$.

Image	**Size [w×h× bpp]**	**Size [byte]**	**Time [ns]**	**Throughput**	
				[Mbyte/s]	**[Mpx/s]**
coastal_b1	$1024 \times 1024 \times 8$	1048576	5252740	199.62	199.62
marstest	$512 \times 512 \times 8$	262144	1320340	198.54	198.54
ice_2kb1	$2048 \times 2048 \times 10$	5242880	21011080	249.53	199.62
india_2kb1	$2048 \times 2048 \times 10$	5242880	21045870	249.12	199.29
pleiades_portdebouc_b3	$1376 \times 320 \times 12$	660480	2213770	298.35	198.90
pleiades_portdebouc_pan	$1400 \times 5504 \times 12$	11558400	38904950	297.09	198.06
p160_b_f	$2048 \times 2048 \times 16$	8388608	21177260	396.11	198.06
sar	$512 \times 512 \times 16$	524288	1334780	392.79	196.39
noise16	$512 \times 512 \times 16$	524288	1371670	382.23	191.11

The data compression throughput is measured via simulation in ModelSim at a clock frequency of 100 MHz. The evaluation is made for the CCSDS reference images presented in [3]. All images are compressed lossless, i. e. the Integer DWT is chosen. Using the Float DWT will lead to almost identical results, since the floating point module has the same timing behavior. The segment size is $S = 128\,\mathrm{Blocks}$. The results for a selection of images are shown in Table 1.

The complete dataset consists of 34 images ($\approx 77\,\mathrm{Mbyte}$). All images were compressed in the hardware simulation. The resulting bit-stream was validated with a functionally identical software implementation of the algorithm. The average data compression throughput for all classes of images is approximately 197.71 Mpxs (198.96 Mpxs for the CCSDS reference images). It is evident that a higher dynamic range of the input images has relatively no impact on the data compression throughput with respect to the pixel rate. The encoding time depends almost only on the spatial size of the input data. This can be explained by the fact that the entropy encoder is optimized for a dynamic range of 16 bit and the wavelet transform module limits the throughput of the system.

For the evaluation of the resource consumption, the internal precision of the integer/floating point arithmetic is set to $24\,\mathrm{bit}$, and the "Integer DWT" as well as the "Float DWT" were considered. The internal resource consumption of the architecture on a Virtex 5 XC5FX130T-1 for a maximum image width of 4096 px is as follows: If only the "Integer DWT" is included in the design and for a given segment size $S = 128\,\mathrm{blocks}$, the usage of slice registers, LUTs and LUTRAM is approximately 34 %, 46 % and 8 %. There is a dependence between the maximum image width and the amount of memory or the number of BlockRAMs. The percentage of used BlockRAMs is 77 % – for $I_w = 1024\,\mathrm{px}$ it is 50 %. A maximum image width of more than 4096 px requires external SRAM. For $I_w = 8192\,\mathrm{px}$, the number of BlockRAMs is approximately 330, which does not fit into the desired FPGA (298 BlockRAMs). The results for an architecture that includes both the "Integer DWT" and the "Float DWT" are as follows: For a given segment size $S = 128\,\mathrm{blocks}$, the usage of slice registers, LUTs and LUTRAM is approximately 46 %, 83 % and 15 %. The percentage of used BlockRAMs is 76 %.

The power consumption of the compression system is necessary to estimate the total power of a higher level component. The relative power consumption per Mpx/s of the "Integer only" version is approximately $20\,\mathrm{mW}/\mathrm{Mpx/s}$ (total 3.873 W; 70 °C junction temperature). The total power consumption of the "Integer and Float" version is approximately $40\,\mathrm{mW}/\mathrm{Mpx/s}$ (total 4.228 W; 70 °C junction temperature).

SUMMARY AND OUTLOOK

A demonstrator was built to test the real-time capability of the system. The architecture was implemented for a Xilinx Virtex-5QV and a single instance is able to compress images at a rate of $200\,\mathrm{Mpx}/\mathrm{s}$ (or $400\,\mathrm{Mbyte}/\mathrm{s}$ for 16 bit images). It operates at a clock frequency of 100 MHz and processes two image pixels per clock cycle. The design ensures that all parts of the system have a high utilization and parallelism. The Virtex-5QV allows compressing images with a width of up to 4096 px without an external memory. Without external memory or additional interfaces, the power consumption of the architecture is approximately 4 W. This example is one of the fastest implementations yet reported and sufficient for recent high-resolution imaging systems. Investigations in the resource and power consumption and in external memory devices show that it will be possible to integrate the design directly onto a focal plane assembly (FPA).

For future developments, it is planned to build FPAs with an integrated image compression module. Not mentioned in this paper, the architecture can be used for multispectral compression, since only a spectral decorrelation technique must be used. Beside the classic scenario of a store-and-download architecture, more advanced application scenarios are imaginable: Change, event or object detection algorithms can be used in conjunction with an image data compression system. The detected areas can be stored in a high quality, while the other areas are stored in low quality. More advanced image processing algorithms for *scene interpretation* or *abnormal event detection* are also conceivable. However, these methods are quite difficult to implement on ground. Since the download data rate are usually much lower than the image acquisition rate, on-board re-assembling of the bit-stream can be done with a software running on a CPU. It is conceivable to use a radiation tolerant version of the Freescale P4080, which has 8 embedded PowerPC cores running at 1.5 GHz.

ACKNOWLEDGMENT

Kristian Manthey would like to acknowledge financial support of the Helmholtz Research School on Security Technologies (HRSST).

REFERENCES

[1] Bernard V. Brower et al. “Advanced space-qualified downlink image compression ASIC for commercial remote sensing applications”. In: *Proc. SPIE* 4115 (2000), pp. 311–319.

[2] CCSDS. *Image Data Compression - Blue Book*. Recommendation for Space Data System Standards. Consultative Committee for Space Data Systems (CCSDS), 2005.

[3] CCSDS. *Image Data Compression - Green Book*. Report Concerning Space Data System Standards. Consultative Committee for Space Data Systems (CCSDS), 2007.

[4] A. S. Dawood, J. A. Williams, and S. J. Visser. “On-board satellite image compression using reconfigurable FPGAs”. In: *Field-Programmable Technology, 2002. (FPT). Proceedings. 2002 IEEE International Conference on*. Dec. 2002, pp. 306–310.

[5] F. Garcia-Vilchez and J. Serra-Sagrista. “Extending the CCSDS Recommendation for Image Data Compression for Remote Sensing Scenarios”. In: *Geoscience and Remote Sensing, IEEE Transactions on* 47.10 (Oct. 2009), pp. 3431–3445. ISSN: 0196-2892.

[6] Catherine Lambert-Nebout et al. “On-board Optical Image Compression for Future High Resolution Remote Sensing Systems”. In: *SPIE Proceedings* 4115 (2000), pp. 332–346.

[7] Li Li, Gang Zhou, Björn Fiethe, Harald Michalik, and Björn Osterloh. “Efficient implementation of the CCSDS 122.0-B-1 compression standard on a space-qualified field programmable gate array”. In: *Journal of Applied Remote Sensing* 7.1 (2013), pp. 074595–074595.

[8] Albert Lin. “Hardware Implementation of a Real-Time Image Data Compression for Satellite Remote Sensing”. In: *Remote Sensing - Advanced Techniques and Platforms* (2012), p. 415.

[9] Albert Lin, Chieh-Fu Chang, Maw-Ching Lin, and Li-Jung Jan. “Field-programmable gate array implementation of Consultative Committee for Space Data Systems image data compression”. In: *Journal of Applied Remote Sensing* 6.1 (2012), pp. 063551/1–063551/13.

[10] Carole Thiebaut, Xavier Delaunay, Christophe Latry, and Gilles Moury. “CNES studies for on-board compression of high-resolution satellite images”. In: *Proc. SPIE* 7084 (2008), 70840F–70840F/8.

[11] Xilinx, Inc. *Radiation-Hardened, Space-Grade Virtex-5QV Device Overview*. 2011.

[12] Pen-Shu Yeh et al. “The new CCSDS image compression recommendation”. In: *Aerospace Conference, 2005 IEEE*. Mar. 2005, pp. 4138–4145.

[13] Guoxia Yu, Tanya Vladimirova, and Martin N. Sweeting. “Image compression systems on board satellites”. In: *Acta Astronautica64* 64 (2009), pp. 988–1005.